

INCLUDES
3 NEW
SCENARIOS
ON CD-ROM

WITH
MULTIPLAYER TIPS
AND STRATEGIES

CIVILIZATION II

THE COMPLETE GUIDE TO SCENARIO BUILDING

JOHN POSSIDENTE

*“What I would
have given to
have had this
book ...”*

—Mick Uhl,
Game Designer



MICRO PROSE



CIVILIZATION II

THE COMPLETE GUIDE TO SCENARIO BUILDING

Other Titles from GW Press

Dominion: Storm Over Gift3—Exclusive Strategy Guide

Duke Nukem 64—Official Strategy Guide

Duke Nukem Total Meltdown—Exclusive Strategy Guide

Oddworld: Abe's Oddysee—Official Strategy Guide

Quake II—Authorized Strategy Guide

Star Trek: Starfleet Academy—Exclusive Strategy Guide

Total Annihilation—Exclusive Strategy Guide

Interactive Strategy Guides (on CD-ROM) from GameWizards

Blood

Duke Nukem 3D

Duke Nukem Atomic Edition

Lords of the Realm

Phantasmagoria II

Redneck Rampage

Shadow Warrior

Star Fleet Academy

Tomb Raider

Order information: Contact The WizardWorks Group, Inc., 2300 Berkshire Lane North, Plymouth, MN 55441 USA. You can also call toll free 1-800-229-2714.

Visit our web site at www.gamewizards.com



CIVILIZATION II

THE COMPLETE GUIDE TO SCENARIO BUILDING

John Possidente

New scenarios by
Antonio Leal-Inglada



GW Press

A Division of GameWizards, Inc.
7085 Shady Oak Road
Minneapolis, MN 55344
www.gamewizards.com

Publisher	Civilization II—The Complete Guide to Scenario Building
Shel Mann	Published by
	GW Press
Acquisitions/Development	A Division of GameWizards, Inc.
Michael Koch	7085 Shady Oak Road
	Minneapolis, MN 55344
Design/Layout	
MK Publication Services	

Civilization II—The Complete Guide to Scenario Building © 1998 MicroProse, Inc. All Rights Reserved. GW Press™ is a trademark of GameWizards, Inc. *Civilization II*, *Civ II*, *Conflicts in Civilization*, *Fantastic Worlds*, *Civilization II Multiplayer Gold Edition*, and the MicroProse logo are trademarks of MicroProse, Inc. All other brand names, product names, and characters mentioned in this book are trade names, service marks, trademarks, or registered trademarks of their respective companies. All other trademarks are the property of their respective companies. No part of this book, including interior design, cover design, and icons, may be reproduced or transmitted in any form, by any means (electronic, recording, or otherwise) without prior written permission of the publisher.

Limits of Liability and Disclaimer of Warranty: GW Press has made every effort to determine that the information in this book is accurate, including the development, research, and testing of the tips, strategies, and procedures to determine their effectiveness. However, the author, copyright holder, and publisher make no warranty of any kind, expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose, with regard to the accuracy, effectiveness, or completeness of the programs discussed or the documentation contained in this book. The author, copyright holder, and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs or the material in this book.

ISBN: 1-56893-904-3

Library of Congress Catalog Card Number: 98-84117

Printed in the United States of America

98 99 00 10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Foreword	ix
Acknowledgments	x
Introduction	xi
Part I: Designing and Building Worlds	1
Step 1: Getting Started	3
Step 2: Create Your Map	17
Step 3: Design Units	57
Step 4: Build a Tech Tree	89
Step 5: Make Improvements	103
Step 6: The Foundations of Civilization	137
Step 7: “Scenario” Means “Situation”	159
Step 8: Schedule Your Events	210
Step 9: Finishing Touches	225
Part II: The Designers’ Notebook	233
Tricks of the Trade	235
The Designers Talk	251
Part III: Multiplayer <i>Civilization II</i>	283
Designing Multiplayer Scenarios	285
Multiplayer Strategies	291
Appendix A: File References	303
Appendix B: Converting Scenarios	311
Appendix C: How to Use the CD-ROM	315
License Agreement	End of Book

Contents

Foreword	ix
Acknowledgments	x
Introduction	xi
Part I: Designing and Building Worlds	1
Step 1: Getting Started	3
What's a Scenario?	4
The Tools You Have	5
Where Do I Start?	6
Thinking About Concepts	7
Creating Files and Folders	11
Step 2: Create Your Map	17
Planning Your Map	18
Building Worlds	27
Customizing Terrain	33
Avoiding Map Problems	50
Using Your Map	54
Step 3: Design Units	57
General Guidelines	58
Units and Advances	60
Defining Characteristics	61
Changing Icons	79
Understanding Unit Sounds	84
Step 4: Build a Tech Tree	89
Some Guidelines	90
Charting the Tree: One Method	91
Defining Advances	94
Changing the Icons	97
Fixed Side-Effects	101
Step 5: Make Improvements	103
A Few Guidelines	104
Changing Things	105
The Rules	105
Modifying Icons	109
Replacing Sounds	112
Effects Reference	114

Step 6: The Foundations of Civilization	137
Cosmic Principles	138
The Text Civilopedia	148
Menus	153
Game Text	154
Step 7: “Scenario” Means “Situation”	159
Defining the Tribes	160
The Game Setup	164
The Importance of Save Files	169
Building Empires	170
The Telling Details	194
Step 8: Schedule Your Events	201
Important Notes	202
Creating the Events File	204
Notes and Quirks	204
Step 9: Finishing Touches	225
Making a Splash Graphic	226
Writing the Intro Text	227
Assigning Music	229
Testing, Tweaking, and Balancing	230
Part II: The Designers’ Notebook	233
Tricks of the Trade	235
Private Tech Trees	236
Limited to a Marginal Victory	236
Unique Units	237
500 Million Years BC	240
Sinking Atlantis	241
Melting the Martian Icecaps	242
Alternate Ways of Winning	242
Invisible Tree Guards	242
The Martian Invasion	243
Gnolam Income Bonus	245
Entrance to Underworld	246
Quests	247
Alternate Scenarios	250

The Designers Talk	251
Mick Uhl	252
Ken Burd	275
John Possidente	278
Part III: Multiplayer <i>Civilization II</i>	283
Designing Multiplayer Scenarios	285
Some Guidelines	286
Balance	288
Multiplayer Strategies	291
General Tactics	292
Scenario Quickies	294
Appendix A: File References	303
Appendix B: Converting Scenarios	311
Appendix C: How to Use the CD-ROM	315
License Agreement	

Foreword

The fact that you are reading this foreword informs me that you have embarked wisely on your career of *Civilization II* scenario design. I only wish that I had been so lucky. Before commencing my first scenario design, I got a ten-minute, information-intensive lecture from Brian Reynolds (the game designer) on what to do. With the mastering date less than a month away, this was all the time he could spare. I learned scenario design simply by trial and error. Two years and seventeen published scenarios later, I am still forced to work the same way. What I would have given to have had this guidebook—and what use I intend to put it to now that I do have it!

John (the author) has profited by his inside track here at MicroProse to pry the design secrets out of all of the people here who have contributed *Civ II* scenarios. He also writes from personal experience, having supplied two well-received scenarios of his own for the *Civ II Fantastic Worlds* scenario pack. You could do worse than to listen to his advice.

Almost all of my favorite design tricks are found somewhere within. Pay attention to the explanation of how to restrict unit construction to particular tribes. This is one of my workhorses and has enabled me to include multitudes of special units in my scenarios. Without it, I wouldn't have been able to give you alien spaceships, great leaders, and all of the fantastic monsters you see in the Midgard and Jules Verne scenarios, among others. Also, read his lucid explanation of the macro language. He explains the complexities of event construction in such a rational way that all of its mystery is peeled away. What I found tremendously useful are the suggestions of creative ways to employ the triggers and actions that he generously interjects throughout his narration. I intend to try out several of his ideas myself. I congratulate him on a very thorough study of a complicated subject, and I shall be astonished if anyone can claim to have not profited from a reading of this book. I humbly admit that I have.

Mick Uhl
MicroProse Software, Inc.
Hunt Valley, MD

Acknowledgments

No one writes a book without help, and I'm no exception. Here's well-deserved thanks to all the folks at MicroProse, past and present, especially: Mick Uhl for his absolutely invaluable help and for encouraging me to turn my half-baked scenario designs into the real thing; Jim Crawley, icon artist extraordinaire, for enhancing the look of the *Civ II* icons used throughout this book; Dave Ellis for his insight into *Civilization II*; Mark Reis for sound advice; Paula Cuneo for taking me seriously; Amy Smith-Boylan for letting it happen on her watch; Antonio Leal-Inglada for his new scenarios, and Dave Osborn for turning lunch into a job interview all those years ago.

As all good editors are, Michael Koch was a true partner in the writing process—maybe sometimes a pain in the neck, but always for the right reasons. His suggestions, criticism, and help were essential to getting the book done and done right.

Naturally, I want to include my family in here—they deserve thanks just for putting up with me all these years: Mom and Dad for everything, and my brother Dave for one unbreakable friendship.

... and Mary, for her patience.

About the Author

John Possidente designed the Age of Reptiles and Mars Now! scenarios—and wrote the manual—for *Fantastic Worlds*. He's been producing MicroProse and SimTex (now MPS Texas) documentation since 1994. John also worked on *Ultimate Civilization II*, and he's slated to help with the design on future Civilization products.

John has also written a few strategy guides, but this is his first book for GW Press. One of these days, he'll finish a novel. Right now, he says, his arms are tired.

INTRODUCTION

This book is not a strategy guide. While it does include lots of useful information and developers' tips that could help you win the MicroProse scenarios, it does not give you explicit instructions on how to beat the various scenarios MicroProse has released for *Civilization II*.

Instead, this book gives you everything you need to know to design and build your own scenarios for *Civilization II*. The three game manuals (and the official strategy guides) introduce you to how the basic tools work—tools such as the Map Editor, the Cheat menu, and the Editors menu—but those books don't cover what you really need to know to build a scenario from start to finish. After reading this book, you'll not only be able to build scenarios as well as any accomplished designer, but you'll be able to take apart any scenario and examine it for yourself (which might also help you win).

To make the learning go as easily as possible—for both the experienced designer and the complete novice—I've broken the design process down into manageable chunks, each of which I explain in detail and without jargon. (When I can't avoid jargon, I try to explain those terms, too.) If you've ever wanted to design a scenario, but been put off by the amount of detail and game knowledge that *seemed* necessary, then this book is for you.

How to Use This Book

The book is divided into three parts, plus three appendixes. Part I, Designing and Building Worlds, is a step-by-step explanation—with plenty of inside information—of how to build scenarios for *Civilization II*. I assume that you've played the game (so you're familiar with the concepts) but not that you're an expert. Part II, The Designers' Notebook, goes beyond basic scenario building with stories, tricks, and advice from the designers at MicroProse. Part III, Multiplayer Civilization II, gets under the skin of the *Civ II* multiplayer game. Appendix A is a handy reference to all the files you would ever want to edit in the course of building a scenario. Appendix B explains how to convert scenarios between the various possible formats. Appendix C tells you how to use the exclusive CD-ROM that comes with this book. The CD-ROM gives you two brand new scenarios designed especially for the book.

Most of the information in this book is equally applicable to the original PC release, the Mac version, both scenario packs, and the multiplayer game. However, whenever I discuss issues that are relevant only if you have a specific product installed, that text is noted in the margin by one of the following icons:

CIV II

This icon marks instructions that are applicable if you have *only* the original version (PC or Mac). If you have either scenario pack (or both) installed, you can skip this info.

SP

Whenever you see this icon, it means that the instructions assume that you have at least one of the scenario packs installed and, therefore, can use the scenario macro language and scenario folders. (Note that *Civilization II Multiplayer Gold Edition* includes the *Conflicts in Civilization* scenario pack.)

FANTASTIC WORLDS

Instructions marked by this icon are only relevant to you if you have *Fantastic Worlds* installed.

In addition to these icons, you'll notice a few other icons that are meant to help you find your way around easier. These alert you to text that provides critical information, warns you about problems, or highlights useful tips.

NOTE

This icon notes a point of special interest.

TIP

If there's a cheat, a shortcut, or a better way that isn't immediately obvious, this icon points it out.

CAUTION

When you see this icon, it means bad things could happen if you're not careful. In those instances when you should exercise caution, I'll tell you what to watch out for.

Now that the fundamentals are out of the way, let's get started on some designs.





PART I

Designing and Building Worlds

This part is chock full of inside information and provides you with step-by-step explanations of how to design and build scenarios for *Civilization II*.

STEP 1

GETTING STARTED

FIRST STEPS ARE ALWAYS EXCITING—and a little scary. Your first foray into game design won't be any different. There's a certain nervous thrill the first time you give one of your scenarios to *someone else* to play—and then wait to see if they'll like it. (I do mean *wait*; evaluating a *Civ II* scenario can take a few hours.)

We're not nearly there, yet. There's a little preparation to go through—crawling before we walk, if you will. Like any other project, a good scenario profits from some thought beforehand. This chapter introduces some concepts for those of you who've never worked on a scenario before, and lays out the different sets of scenario-building tools you might have at your disposal. After that's all taken care of, we begin the scenario design process in earnest.

Let's go!

What's a Scenario?

With the random map generator and all, *Civilization II* has a lot of what industry pros call “replay value”—which means you can play it over and over until all your hair falls out and continental drift affects air fares, and every game will be different. That's great, but because of this, your average *Civ II* player probably doesn't know much or care much about scenarios. If you're part of that majority, keep reading. (If not, keep reading anyway; you might learn something.)

There's nothing wrong with randomly generated worlds. However, a pre-designed situation can offer an immediate challenge and an intensity that starting the game from scratch in 4000 BC never could. In *Civ II* terms, a designed situation is called a *scenario*. The original game included a couple of scenarios that were no more than situations. The WWII and Rome scenarios each used standard units, cities, and advances, placed to approximate a moment in history that would be interesting to take part in. When you played the scenario, you took command of one of the major forces of the time, and you stepped hip deep into the situation. That's what makes some scenarios different from the standard game—when you start, you're already in trouble.

NOTE In fact, the WWII scenario has one peculiarity that no other scenario to date shares—its own, unique artificial intelligence (AI). Why did this scenario need the extra programming? It was the only reliable way to ensure that the computer-controlled civilizations would behave like their historical counterparts. Normally, you can accomplish that by setting up the scenario correctly, so don't worry that you can't make a new AI for your scenarios.

Later, while the designers were working on the first group of scenarios, *Conflicts in Civilization*, they decided that the standard game was much too limiting. Playing situations was interesting and enjoyable, but introducing new units, advances, and improvements in every scenario would widen the possibilities and be much more fun. That's when the designers started getting their fingers into the files—modifying the units and rearranging the technology tree. Don't get the idea that this wasn't planned; *Civilization II* had the capacity for scenarios built into it from the start. For *Conflicts in Civilization*, this capacity was substantially expanded, but it wasn't until *Fantastic Worlds* that the scenario designs went so far that they forced MicroProse to seriously reengineer portions of the game.

So what is a scenario, then? A scenario is a game that you design, a game based on the *Civilization II* rules, but different enough from the standard game that it's fun even for people, who are a little bored with the standard game. The question that naturally follows is: Well, what can I change? The answer depends somewhat on which of the packages you have installed. With the right tools, you can change *almost everything*.

A note of caution is in order for the potential entrepreneurs among you. Designing a scenario isn't the same thing as designing a whole new game. MicroProse created and owns *Civilization II*, and you can't create or play a scenario without it. For this and other reasons, you can't legally sell your scenarios without written permission from MicroProse. (Just in case you were thinking along those lines.)

CAUTION

The Tools You Have

What tools you have at your disposal depends entirely on what *Civilization II* products you have installed. Both add-on packs—*Conflicts in Civilization* and *Fantastic Worlds*—include utilities that give you more power over your worlds or make existing possibilities easier to realize than if you have only the original game.

Civilization II

The original *Civilization II*—both PC and Macintosh versions—has everything you need to create scenarios. However, if you want to change things substantially, you'll end up making a second copy of the game. That means you'll need as much free hard drive space as you did when you installed the game. If you want to change the look and sound of the game, you'll need programs to edit GIF and WAV files. You also end up doing a lot of the work yourself. (Personally, I prefer this method, because it forces you to learn your way around the files.)

GIF Editors

GIF (Graphics Interchange Format) is a proprietary graphics format developed by CompuServe for bitmap graphics of up to 256 colors. There are several programs that enable you to edit GIF files—all of them (presumably) licensed by CompuServe. Many of the most popular graphics programs offer the ability to edit GIF files (or to convert them into a format the program can edit, then back into GIF format).

If your favorite editor doesn't handle GIF format, I'd suggest buying *Fantastic Worlds* and using the built-in editors; it's likely to be less expensive than any graphics program (unless you can find a shareware or freeware GIF editor somewhere).

WAV Editors

The WAV format is one of the most widely used formats for sound files. All of the most popular sound programs offer the ability to edit WAV files and, therefore, the ability to convert them into a format you can use with *Civilization II*.

None of the *Civ II* products gives you the ability to edit or create WAV files, but even older versions of Windows included a WAV utility. If you look in your Main or Accessories windows (or the corresponding sections of the Start menu in Windows 95), you'll probably find that you already have a WAV editor.

Conflicts in Civilization

Conflicts in Civilization, the first set of scenarios, introduced separate scenario folders, which enabled the designers to have greater latitude in editing the rules and graphics files. *Conflicts* also brought the concept of *events* to scenarios, and it equipped world builders with a rudimentary macro language with which to create them.

Frankly, both the scenario folders and events are great. Now that I've used both, I wouldn't want to design a scenario without them. *Fantastic Worlds* also included folders and the events macro language, so if you have either scenario pack installed, you can use both.

Fantastic Worlds

Fantastic Worlds, the second scenario pack, expanded the scenario macro language and added a few extra "slots" for new units and advances. It also supplied a menu of editors that enable you to do most of the grunt work of scenario creation inside the game, rather than in your text and GIF editors.

MicroProse also—without mentioning it in the manual—snuck in a small library of icon art you can use. Most important from the scenario-building standpoint is that *Fantastic Worlds* also changed the format of a few of the standard files.

NOTE

Scenarios created with *Civilization II* and *Conflicts in Civilization* will work just fine if you have *Fantastic Worlds* installed, but scenarios created with *Fantastic Worlds* will *not* work with those earlier versions. A lot of the files have changed format. It's possible to convert a scenario to the earlier format, but it's not what you'd call easy. See Appendix B: Converting Scenarios for the scoop.

Where Do I Start?

If you sit back and think about the prospect of creating a scenario, it's easy to be daunted by what seems like an enormously complicated project. There are so many things to do, so many details to take care of, and so many interrelations to keep track

of, that you might be tempted to throw up your hands and just forget the whole thing. Relax. It's nowhere near as difficult as it seems. Some of us even think it's fun.

At the beginning, though, you're faced with the same question that stymies many creative people: Where do I start?

Start with the fun part. Work on whatever part of your scenario you find most interesting. It doesn't matter if you feel like you're starting in the middle, because the process of designing a scenario isn't like climbing a ladder—you don't go step by step in a specified order. The beginning is when you come up with an idea, the end is when you decide that you're done. Everything else is “the middle,” and you can do it in whatever order you like.

In this book, I present the scenario creation process in a sequence that has worked for me and generally helps you avoid backtracking and extra work. However, there's no reason to believe that my way of doing things is the only way. The best method of designing a scenario is to do things in the order in which they catch your interest. (You'll generally do your best, most creative work on something you're excited about.) As long as everything gets done, the order you do things in doesn't really matter too much. Just remember, designing a scenario is a game, not a chore. Treat it like a game, and you'll enjoy it.

NOTE

Thinking About Concepts

Now, when I say to start with the fun part, what that means for most people is coming up with an overall guiding idea for the scenario. If you're reading this book, chances are you already have an idea for a scenario, so the first step is already out of the way.

It doesn't matter if your idea is complete or just a half-baked notion. For example, you could say to yourself, “I'd like to do a scenario that's a detailed historical re-creation of the Trojan War,” or, “I want to do a scenario with elves in it.” Both are perfectly good concepts. What's important is that you don't just start flailing around aimlessly. (You could do that and come up with a great scenario, but the odds are against it.)

The ideas we've started with at MicroProse have run the gamut. Mick Uhl plans out his historical scenarios very carefully (see Figure 1-1), but when he began work on the Alien Invasion scenario, his whole concept was, “I want to do something along the lines of *War of the Worlds*.” When Bob Abe started work on the X-COM: Assault scenario, he had a very specific plan for a fast-paced firefight that was hard to win. My own Mars Now! scenario, on the other hand, wasn't even conceived as a scenario—it started out as an idea for a map.

Figure 1-1: Mick Uhl's historical scenarios were carefully planned.



Some Ideas Are More Equal than Others

While it's true that any idea can become a scenario, some concepts are better suited to becoming fun *Civ II* scenarios than others. How can you tell the difference? That's tough to say, but there are a few guidelines you can follow, including

- ▼ **Conflict is vital.** It's important that the scenario concept includes—or at least allows for—conflict between groups, since conflict is the basic nature of *Civilization II*. This doesn't mean that every scenario has to be a war. Conflict need not be military in nature, but war must not be out of the question. (Remember, world peace is one of the goals of the game, so it shouldn't be too easy to achieve.) Usually, the natural pressures of empire expansion and limited resources and territory will create the conflict you need. Place the different civilizations close enough to one another, and the AI does the rest for you. Even in the World of Jules Verne scenario, in which the primary goal is exploration, war is almost inevitable.
- ▼ **Overwhelming odds are risky.** Unless you're setting up a very specific situation (like the one in *X-COM: Assault*, in which there is only one civilization you intend the human player to control), it's a bad idea to have any single civilization overwhelmingly more powerful than the others. You can't always predict the way the

computer-controlled civilizations will act, and the balance of power you expect to evolve might not evolve.

For example, when Mick was designing the Alien Invasion scenario, his original idea was that the aliens would be extremely powerful, and that that would force all the other civilizations to unite in alliances against the invaders. Trouble was, the AI didn't see it that way. The computer-controlled civilizations would declare war on each other or on the human player almost as often as they would attack the Hodad invaders (see Figure 1-2). The first version of that scenario was virtually impossible to win. It took lots of fine-tuning to get the power balance just right. No scenario should be easy to win, but it must be *possible* for a good *Civ II* player to win without too much luck.



Figure 1-2:
In early versions of the Alien Invasion scenario, the aliens were almost unbeatable.

- ▼ **Foregone conclusions are boring.** No matter how realistic you want your scenario to be, if you allow the human player to rule the Confederate States of America, there must be at least a slim chance that they'll win the war. If you already know which civilization is going to win, you should rethink your concept. Even in the Atlantis scenario, the Atlanteans have a good chance to survive the destruction of their home islands. Predestined endings sometimes make for good literature, but they make lousy scenarios.

- ▼ **Good ideas multiply.** One way to recognize that you might have latched onto a really good, rich concept is that it immediately sparks all sorts of inspirations in your head. Ideas for units, advances, and other parts of the scenario spring to mind one after another like falling dominoes. The Age of Reptiles scenario makes a perfect example. As soon as the idea for a dinosaur scenario popped up, the unit designs were obvious, and so was the Pangaea-like map (see Figure 1-3). It was a short mental step to the technology tree based on evolution. The hard parts were things like coming up with city improvements and Wonders of the World that made sense in a prehistoric, reptilian setting.

Figure 1-3: Most of the scenario design followed easily from the dinosaur concept.



When all those ideas for bits of the scenario design strike, don't discard them. Don't say to yourself, "It's too early to be thinking about that." Keep all the ideas, especially the ones that you're excited about. Start working on them right away. *That* is what I mean when I say, "Start with the fun part."

Remember, designing a scenario shouldn't be work—it's play. Focus on the things you want to do, and you'll make fast progress. If you focus on the parts of the process that you find tedious and difficult, you'll get bored and never finish. Yes, you'll probably need to come back to the tough parts eventually, but they'll wait. By the time you get around to them, they might have become interesting.

First, though, you must set up the design infrastructure—the folders and files for your scenario.

Creating Files and Folders

You can design on paper until the cows come home, and it's a good idea. Especially when working on the advances—the tech tree—the smartest designers work out their plan completely before they sit down and start mucking about in the *rules.txt* file. (You'll need a big sheet of paper for that tech tree design.)

When you do finally decide to begin working on your scenario in earnest, there are a few preparatory steps you should take first. How you take those steps and even what steps you must take depend (as many things do) on which of the *Civilization II* products you have installed.

Setup with *Civilization II*

If you have only the original game (PC or Mac version) and neither of the scenario packs, the setup process for creating a scenario can be quite a bit more complicated. When the original *Civilization II* was created, the concept of what constituted a scenario was simple: take all the existing parts of the game, use them to create an interesting situation, and add an introductory text display to explain the circumstances to the player. That was it.

To that end, the game was engineered so that when the player chooses to load a scenario, the program looks in the *Civ II* base folder (the folder to which you installed the game)

Mick Uhl's Basic Design Guidelines

Mick Uhl is the primary game designer behind both of the MicroProse scenario packs. Among other things, he created all the scenarios for *Conflicts in Civilization* and the majority of those in *Fantastic Worlds*. In my book, that makes him a recognized expert. Asked about scenario concepts, Mick says, "I alter my criteria slightly, depending upon whether the scenario is historical or not." According to him, an historical scenario should:

- ▼ **Cover a well-known period.** The American Civil War generates more interest than the Thirty Years War, however superior the latter is as scenario material in all other respects. Hence, the Civil War became a scenario and the Thirty Years War did not.
- ▼ **Have at least three protagonists**, two of which are evenly matched. Scenarios with two civilizations are practically impossible to balance.
- ▼ **Cover a long period of time.** Scenarios of short duration, such as WWII, are inevitably pure war games. You need time to develop technology and city improvements.
- ▼ **Be empire-building in nature.** The game can't indicate victory in any other way.

For *ahistorical scenarios*, Mick tries to follow the last three guidelines above, while also considering that:

- ▼ **The subject matter is not that important**, as long as it has the potential for a rich military environment.
- ▼ **The progress of units should be interesting.** You want to be able to develop a time-line of interesting military forces with some kind of "super unit" at or near the end of the tree.
- ▼ **The protagonists should not be identical.** Each protagonist must have something in its identity that easily distinguishes it from all others. Mick's Midgard scenario is the best example of this.

Mick doesn't consider technology or city improvements as criteria, as the best he (and we) can do with them is rename them.

for two files. The first is the actual scenario file—*scenname.scn* (where “scenname” is the name of the scenario). This is essentially a saved game under a different extension (*.*scn* instead of *.*sav*) and with scenario parameters defined. The second file is the introductory text—*scenname.txt*.

Under this setup, the possibilities for really creative scenarios can be somewhat limited. If you want to make radical changes to the game—changing the way things look, editing the rules, and so forth—your changes will affect every game you play. Why? It’s because you must edit the game files—files other than the two the program looks for—to make serious changes, and those files are used as the basis of every game of *Civilization II* you play.

For that reason, you should make a complete copy of *Civilization II* on your hard drive if you plan to make substantial changes to the game. (As long as all the copies remain on the same computer, it’s technically not software piracy.) This is one strategy for building multiple scenarios—you can keep one copy of *Civilization II* untouched and pristine, while you build scenarios using an alternate copy.

NOTE

Take heart from the fact that no matter how many changes you make and no matter how big a mess you make of things, you can always fix everything simply by re-installing the game.

You can still create awesome scenarios, but if you want to make radical changes to the game, you must take some radical steps. Here are the things you can do without significantly affecting the way *Civilization II* plays your non-scenario games and other scenarios:

- ▼ Create a unique map
- ▼ Add 3 new units
- ▼ Add 3 new advances
- ▼ Set up an exciting situation
- ▼ Change the names of all the civilizations and leaders

- ▼ Specify the diplomatic, exploration, and technological status of every civilization
- ▼ Erase or restore all the villages (“goody huts”) on the map
- ▼ Create an introductory text message

If this doesn’t seem like much to you, take a look at the Rome and WWII scenarios. There’s a lot you can do with the game’s basic tools.

If you decide after all that you want to pursue large-scale changes to the game for your scenario, you have two options (assuming that you do not want to go out and get one of the scenario packs):

- ▼ Change the way the game works, or
- ▼ Copy the entire game and change the way that copy of the game works.

Whichever you choose, you must make your choice before you begin.

Sample Concept for a Potential Scenario

A potential scenario might be based on the brilliant concept of replacing the human civilizations with colonies of insects. Right away, some design ideas come to mind. In this scenario,

- ▼ We could have bees, wasps, termites, ants, and maybe cockroaches.
- ▼ Cities would look like hives, termite mounds, and anthills. Cockroach cities would be tough to represent. Maybe we shouldn’t include roaches.
- ▼ The natural resources could be things that insects need, like pollen for the bees, wood for the termites, and edible plants for the ants. What do wasps need? Do they eat other insects? Have to look that up.
- ▼ Units would include Workers and Drones, and maybe a unique Queen for each colony, but the Queen should probably be unable to move—or else be *really* slow.
- ▼ If the bees and wasps are all air units, they wouldn’t have any settlers and the other colonies could never attack them. That might be realistic, but it also might give the bees a heavy advantage or disadvantage.
- ▼ Naval units would be fairly limited. Maybe the ants and termites could build little leaf boats.
- ▼ Which one of the government types would best represent the insects’ collective society—Communism, Monarchy, or Fundamentalism?
- ▼ If you have one of the scenario packs installed, we could include events that destroy the whole colony when the queen—and presumably the capital city—is captured.

Sounds like it might work, although there are a few pitfalls lurking. The potential for research seems rather slim. There’s also no obvious way to design the map of the insect world. Well, we’ll work those issues out when the time comes.

Setup with Either Scenario Pack

If you have installed either of the scenario packs (or both), the setup that enables you to create multiple scenarios with all sorts of excessive modifications to the game seems almost trivial, when compared to what folks must go through when they have only the original version of the game. That's because of a little addition Kerry Wilkinson (the programmer for *Conflicts in Civilization*) made: the *Scenario* folder.

In the process of designing *Conflicts in Civilization*, the development team realized that the two-file scenario scheme would severely hamper their freedom to create. (If you don't understand that statement, it's because you didn't read the previous section.) To get around this limitation, Kerry reworked the game so that every scenario and all the files that were changed to build it could be housed in a separate folder. He instituted the *Scenario* folder—a subfolder of the *Civ II* base folder—to hold all the various individual scenario folders.

This enables you to build a scenario without making any unwanted changes to the way normal (non-scenario) *Civ II* games play. So, here's what you need to do to prepare the way:

- 1 **Create a subfolder in the Scenario folder.** You must create a folder in which to keep your scenario files. Make this folder a subfolder of the Scenario folder that is in the directory to which you installed *Civilization II*. This isn't just a way to keep organized. When the game program tries to load any scenario from this folder, it looks in this folder for all the files it needs. Any file the program finds here is used in place of the original *Civ II* file (with lots of exceptions, which I'll go into later). For every file the program does not find here, it uses the original file. You must limit the folder name to eight characters or less.
- 2 **Create a *Sound* sub-subfolder in your scenario subfolder for sound files.** If your scenario uses the usual *Civ II* sounds, this is not necessary, but if you have even one new sound or rearrange the existing ones, you need a sound folder. The sound folder must be named *Sound*. This works just like the scenario folder; any sound file the program finds here is used in place of the original, and for every one the program does not find here, it uses the original.
- 3 **Copy the files you intend to modify from the *Civ II* base folder into your scenario folder.** Don't worry if you don't know for sure which ones you need yet; you can always copy more over later. The first time you build a scenario, you might not copy over any files at this point. The second time around, though, you'll have a better idea which ones you need.

What method you use to take these steps depends on which of the scenario packs you have. As long as you don't have *Fantastic Worlds* installed, you can simply use Windows or DOS to create the folder for your scenario and the subfolder for sound files. Then you can copy over files in the same way you would copy any other files.

A Note about Setup with *Fantastic Worlds*

The automated scenario generation tools included with *Fantastic Worlds* actually make this particular step a little bit more complicated. You can have the game create the necessary folders for you, but you must be in the middle of a game to do so, and the game you're in the middle of is turned into a scenario. This makes it more convenient, generally, to create your map first, then start a game using that map, and then immediately create the folders (during Turn 1).

Why is that more complicated? There are some decisions you make and some data loaded when you start a game, especially if you choose to customize things (which you normally will when creating a scenario). All the game settings (the intensity of Barbarian attacks is the best example) are retained in the scenario and it can be very difficult to change them later. (I'll note which they are as we go through the process.) If you haven't already decided on (or designed) the settings in question, it can be inconvenient to lock them in this early in the process. However, for those of you who already know what you want early on, this minor inconvenience is no inconvenience at all.

Regardless, here's how you create the folders using *Fantastic Worlds*:

- 1 Open the **Editors** menu (assuming that you already have a game running).
- 2 Select the only available option, **Toggle Scenario Flag**. (The rest are grayed out at this point.)
- 3 Confirm that you want to activate **Cheat Mode**.
- 4 Read the dire warning, then click to get rid of it.
- 5 Type in a name for your scenario folder. (You only have eight characters, so make it short.) Note: You *cannot* overwrite an existing scenario folder.

As you continue through the process of creating the rest of your scenario—units, advances, and all that—the *Fantastic Worlds* editors copy the necessary files over from the base folder for you. There are a few files that you can't edit using the editors, and if you want to change the things these files control, you'll need to do it the hard way. (Okay, it's not really that hard.)

TIP

Even if you have *Fantastic Worlds* installed, you can still create your folders the old-fashioned way. However, if you plan to use the editors to edit other portions of the scenario, you should not copy over any files except those that the editors don't affect. (Turn to Appendix A: File References to find out which files those are.)

So far, so good. Now that we've covered the basics, we're ready to get into the meaty part and start building the scenario. ✂

STEP 2

CREATE YOUR MAP



KAY, YOU'VE WEIGHED THE PROS AND CONS of your scenario concepts, and now you have a pretty good idea of what sort of scenario you're going to build. You know what tools you have, and you've created the necessary folders. Now what? Well, if you follow the lead of the designers at MicroProse, you'll start with the map. Is it absolutely necessary to make the map first? No, of course not. Even if you have no map at all, you can do everything except actually put together the scenario situation. However, taking the time to make a map (and it can take some time) can save you lots of time and frustration later. As Mick Uhl (who has designed more published *Civ II* scenarios than anyone else) puts it,

I build the map first primarily to get a handle on the scale of the game. It's easier to adjust other things to the map than vice versa. Also I can't begin to use the cheats until the map is done, and I prefer to be able to make changes immediately when I think of them instead of writing them down for future implementing.

Planning Your Map

It's mighty convenient to have your map (or something resembling it) handy when you need to test parts of your scenario—how units and cities look against the terrain, whether your new units have reasonable movement allowances, whether or not your events work the way you expect them to, and that sort of thing.

FANTASTIC WORLDS

If you're working with the scenario construction tools in *Fantastic Worlds*, making your map first also enables you to start up a game, then create the necessary scenario directories using the Editors menu.

The first step in making your scenario map is also the biggest step; you must have an overall plan. Some scenario concepts, especially the historical ones, are specific enough that your map plan is obvious. The American Civil War map was easy to design (see Figure 2-1), and so was the map for Mars Now! (Easy to design does not necessarily mean easy to build.)

Figure 2-1: The American Civil War could only be fought on one map.



Even with a very specific concept, however, knowing when to stop can be difficult. The Samurai map obviously needed to center on Japan, but Mick had a tough time deciding how much of mainland Asia to include. If he had included too much, he would have risked making the mainland too important, and thus shifting the strategic focus of the game. If he had included too little, the outside influences that were so important historically might have had no real effect on the warring houses. Your map should include exactly what it needs and no more.

There will also be times when you're sitting and staring at a blank map filled with nothing but ocean (and all those silly fish and whales), racking your brain without a clue where to start. What I do then is play—fiddle around with the Map Editor long enough, and something will start to take shape.

If you're *really* stuck (or just lazy), there are some shortcuts you should know about.

A Few Shortcuts

Personally, I love starting with a blank ocean world and painting in the land. (Why else would I have spent several weeks—no kidding—making an accurate map of the planet Mars, square by square?) Not everyone feels the same way, though. Even if you know exactly what you want to do, map-wise, building a world from scratch can seem like a chore—and nothing about this process should be so painful.

For those of you who don't believe that making a map will be fun, here are some ways to avoid most or all of the work:

- ▼ Borrow a map
- ▼ Use a random map
- ▼ Simplicity

Borrow a Map

You can avoid map-making entirely simply by using one of the maps that MicroProse has generously supplied. The original version of *Civilization II* included seven premade maps (one of them is shown in Figure 2-2), all of which reside in the base directory and are described in Table 2-1. You can also borrow the map from any saved game or existing scenario.

Figure 2-2: An
Odysseus scenario?
The map is already
done.

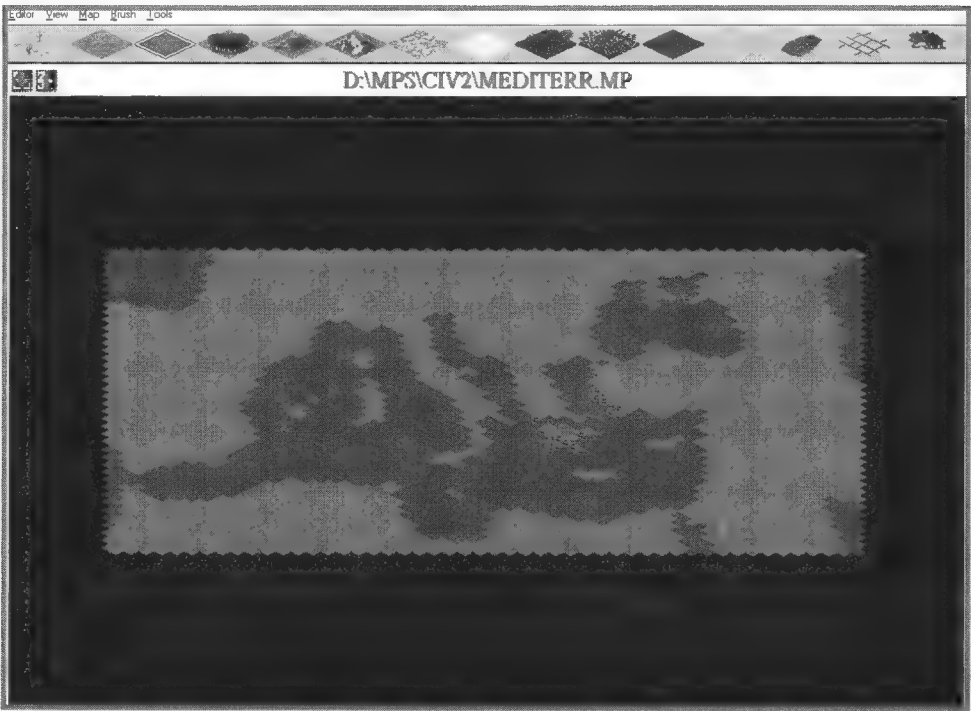


Table 2-1. Existing Maps

MAP FILE	CONTENTS
<i>Europe.mp</i>	This is the map used in the WWII scenario, but it would be just as well suited to almost any period in European history. (Remember the Thirty Years War idea?) It's a large (75 x 120) map that includes all of Europe, the Scandinavian peninsula, most of the Atlantic Ocean, a little of North America, a token Greenland, northern Africa, a good deal of Asia, and most of the Arab world.
<i>Greece.mp</i>	This is a medium-sized (50 x 80) map of the eastern Mediterranean and points east all the way to India. The map for Alexander the Great is based heavily on this one. If you want a map that's almost perfect for modern scenarios of mideast political strife, you've found it.
<i>Mediterr.mp</i>	The Rome scenario is set on this map of the Mediterranean Sea. It's a medium-sized (50 x 80) map, but manages to include southern Europe (including Greece, Italy, and most of Spain), northern Africa (including half of Egypt), and the western part of the Arab world. If you believe, as many archaeologists do, that Minoan Crete was the source of the legends of Atlantis, you might want to build your own version of the Atlantis scenario.

<i>Pacific.mp</i>	Tired of the Eurocentric maps? This one centers around the Pacific Ocean. It's a large (75 x 120) map that includes Japan, Australia, the Korean peninsula, the western third of North America, Hawaii, a large eastern portion of Asia, all the larger Pacific islands, Alaska, the Bering Strait area, and a bit of northeastern India. For <i>Conflicts in Civilization</i> , we thought about including a WWII-era historical scenario based on this map, but we had already done one WWII scenario, not to mention 1942 <i>The Pacific Air War</i> . This area is sometimes called the "Ring of Fire" because of its tremendous volcanic activity.
<i>World.mp</i>	Those of you who played the original <i>Civilization</i> might remember the Play on Earth option. If <i>Civilization II</i> had that option, this is the standard map of Earth that you would end up playing. This is the largest of three Earth maps that the game installs; it's 75 x 120. You can use this one if you want to build a nice, big scenario and include the entire world in the action, or if you want to start with a modified version of the Earth (after the polar caps melt, for example).
<i>World_m.mp</i>	If the large Earth map is a bit too much for the scenario you have in mind, but the small one is too cramped, try this medium (50 x 80) version.
<i>World_s.mp</i>	This is the smallest of the three maps of Earth. It's a 40 x 50 map, so the terrain squares are numbered from 0,0 through 39,49. In fact, this map is so small that when you load it, you'll notice that the World Map looks blocky, and Spain is one square! Maps this size make for wild, fast games.
<i>Tutorial.sav</i>	When we wrote the tutorial (in the original game manual), we generated a random map and worked from there. This map has no special features, but if you like it, go ahead and use it.

When you install *Fantastic Worlds*, you get a few additional maps: *Battle.mp* is the map used in the Battle of the Sexes scenario; *Eastus.mp* is the map from the American Civil War scenario (the scenario itself is part of *Conflicts in Civilization*); *Namerica.mp* will be familiar if you've played the New World scenario; and *Samurai.mp* is taken from the scenario of the same name.



When you open the Editor menu in the Map Editor and select Load Map, you should notice that files with the *.mp (map) extension are not the only ones listed. You can also borrow a map from any saved game (any file with the extension *.sav). That means that you can grab the map from any game you've played and kept, and you can use the map from any scenario you have installed (like the one in Figure 2-3).

Here's how you would go about borrowing the map from your favorite scenario:

- 1 First, load the scenario as if you were going to play it. It doesn't matter which civilization you choose to lead.
- 2 As soon as your first turn begins, save the game. (Press **Ctrl-S** or select **Save Game** from the **Game** menu.) It doesn't matter what you name the saved file, as long as you can remember the name for a minute or two.
- 3 Once the game is saved, exit *Civilization II*.
- 4 Start up the **Map Editor** and select **Load Map** from the **Editor** menu.
- 5 The game you just saved should be in the list. (If it's not, you need to find the folder you saved it into.) Load the saved file.
- 6 Now use the **Save Map As** option on the **Editor** menu to store the world in a map file. You can give it any name you like, but always use the *.mp* extension.

Figure 2-3: You can use any of the scenario maps.



Once you've got a map file to work with, you can use it as is or make whatever modifications are necessary for the purposes of your scenario design. If you make changes to one of the premade maps, you should save your new map under a different name—even if it's in a scenario folder—to avoid accidentally overwriting the original. If you borrowed a map from a saved game or a scenario, you needn't worry about that, because there is no original map file to destroy.

Use a Random Map

Another way to avoid most if not all of the map-building work is to use *Civilization II*'s random map generator. This is an imperfect solution if your scenario requires a particular type of map, but for those scenario concepts that don't really inspire a map idea, this might be your best strategy. Always keep in mind that the random map you generate is only the rough draft from which you will fashion the final map—unless you get lucky, that is.

One serious benefit to having the Map Editor generate a map for you is that it will never make an illegal map or a map that the game will have problems with.

TIP

You don't need to start a game of *Civilization II* to make random maps. The Map Editor enables you to do so, too, and it's quicker. Select the Generate Random Map option on the Map menu. You'll be prompted to make the usual set of decisions, including:

- ▼ World Size
- ▼ Landmass
- ▼ Land Form
- ▼ Climate
- ▼ Temperature
- ▼ Age

World Size determines the scope and—to some extent—the complexity of the scenario. (It's difficult to build a multifaceted, precariously balanced politico-economic situation on a small map.) This is the only one of these options with permanent effects; you cannot change the world size later, so think long and hard about this one. For most scenarios, one of the default choices works fine. If you have a specific map

design in mind that requires a custom map size, you probably won't be happy with any of the random maps, anyway.

Landmass has surprisingly little effect. However, if you do not want the civilizations in your scenario to come into contact too early, it helps to limit the use of Triremes by setting a *Small* landmass. Except for that consideration, the general rule is that the more water you have, the more important naval units will be during the game.

Land Form also has a lesser effect than you might think. No matter which option you pick, there will be both continents and islands. (Otherwise, normal random-map games could be less than fun.) This setting only affects the relative distribution of the two land forms. If you want to slow or limit the growth of civilizations while at the same time making naval units more important, use plenty of islands.

Climate avoids extremes, but can still give you an interesting map. Both the *Arid* and *Wet* options generate a world that's lacking in resources, thus making the scenario tougher and more competitive. You won't get any desert worlds, but you can build those yourself.

Temperature has relatively minor effects. This controls the relative distribution of the types of wasteland terrain—Tundra and Glacier versus Desert and Jungle. As with Climate, you'll get no extremes, but if you want big polar caps or a post-global warming world, this can save you a little map work.

Age controls the size of areas of terrain. If your scenario needs multiple large tracts of the same type of land (like those in the Age of Reptiles), try a *Young* world. If the development of cities is important to your scenario, however, you might be better off with one of the other options.

When the map is done, take a quick look. If it's clear at first glance that it isn't what you want, don't waste any time, just go ahead and generate another one. The settings you chose should have been saved as the temporary defaults, so you can breeze through the generation process. Expect to go through quite a few trials before you get what you want. Figure 2-4 shows an example of a map generated with the random map editor.



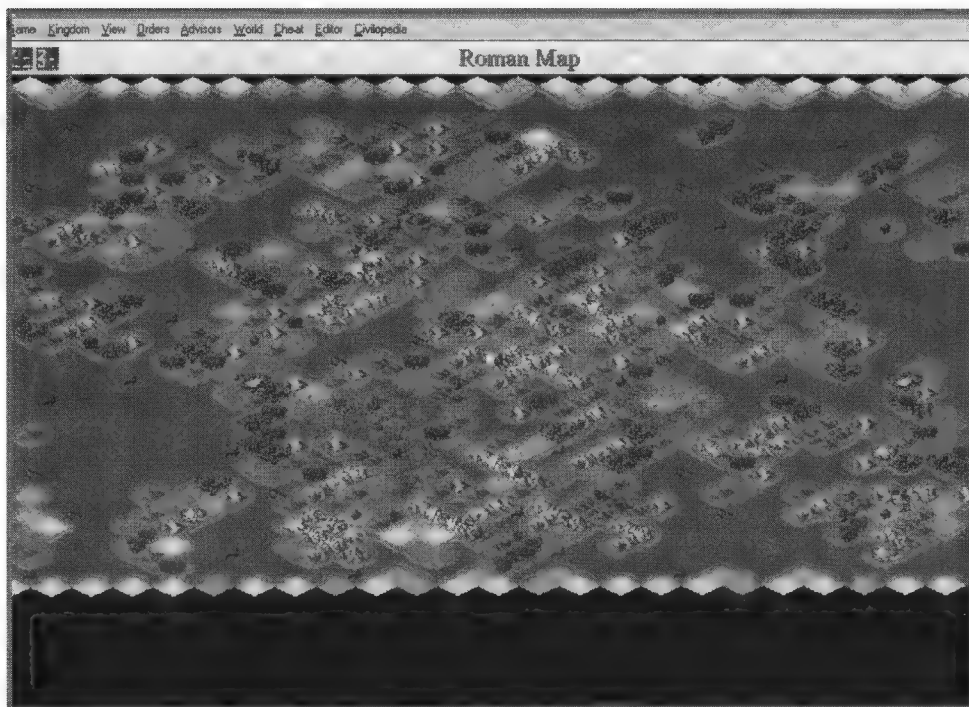


Figure 2-4: Here's a Small, Small, Varied, Arid, Cool, Old world.

When you *do* finally get a map that might work, take a good long look over it. Figure out how well it fits the needs of your scenario concept and—more important—how much editing and tweaking the map needs. Make sure that there are enough workable sites for cities—the scenario is no fun if all the competition dies off on its own. If you still like it after looking at it from all these angles, save it and move on to the section “Building Worlds” in this chapter.

Simplicity

The map-making shortcut strategy I've seen used most often is probably the least likely to result in an interesting map, but that doesn't mean it's useless. If you keep the map simple, building it is not difficult.

If you're not careful, you'll wind up with a really amateur-looking map. For example, let's say your scenario involves four competing civilizations. One simple map design would be to create a continent for each nation, plop the starting position of a civilization down in the middle of each one, and call it a map. It would work, but what's interesting about it? Why would you (or anyone else) want to play on that map?

The difference between a dull simple map and an interesting simple map is in the details. Take that same map and add lots of randomly placed islands and some island chains—but not so many that a Trireme could move from one continent to another. Put natural harbors, oddly shaped peninsulas, and an isthmus or two on the big continents, and make one or two smaller, empty continents. Add an inland sea to one or two of the big land masses. Now move the starting positions of your civilizations nearer the edges of the continents. Add rivers—not too many, but enough to allow plenty of potential city sites and early travel routes. Now, you’re moving toward an interesting world.

Take a look at the map for the Labyrinth scenario (see Figure 2-5). (That’s one of the scenarios on the disk that came with this book.) It’s a simple map; there are only two continents and one ocean between them. The only real detail is in the coastlines, the islands, and the distribution of terrain types. In this scenario, the map is not meant to be as important as in most games—conquest and expansion are not the primary goals. Still, there *are* details; the map would be lifeless without them.

Figure 2-5: On a simple map, the details make the difference.



Building Worlds

That's enough shortcuts and time savers. Assuming for the moment that you're like me and prefer to build a map from scratch, let's get right into it.

Of course, making a map “from scratch” doesn't necessarily mean that you *invent* the map. Creating *Civ II* worlds based on maps of real places is one of the staples of scenario design; you can't design a historical scenario without doing it.

The Grid System, Briefly

Those of you who want to start on your map and don't really care about the map grid numbering system can go right ahead and skip this subsection. It's not really necessary that you understand how the grid works to make a great map. However, if you want to make a map that's reasonably accurate—for an historical scenario or (like Mars) reflecting an actual place—it doesn't hurt to know how to find reference points on a *Civ II* map (maybe even enough to fake longitude and latitude).

If you intend to put any location-specific events into your scenarios—and many of the possible events require you to specify locations or ranges of locations—knowing the difference between the Map Editor map grid numbering system and the in-game grid numbering system can help prevent errors.

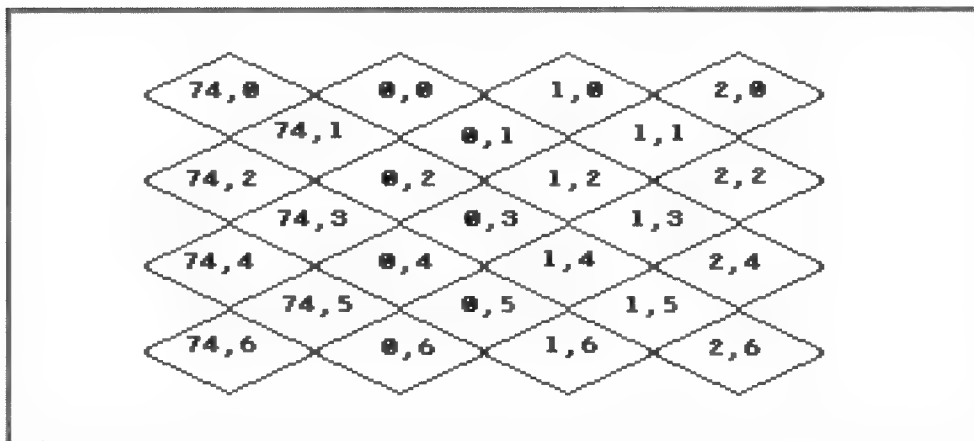
SP

For those of you who are not too familiar with map coordinates, the *x* (horizontal) coordinate is the first one and the *y* (vertical) coordinate is the second one. So, for example, the square at (5,9) is fifth from the left—count starting with zero—and ninth from the top.

NOTE

The most important thing to know is that the grid in the Map Editor and the grid in the game are not the same. They're the same type of grid, but the numbers are off. Take a look at Figure 2-6; that's the numbering system the Map Editor uses.

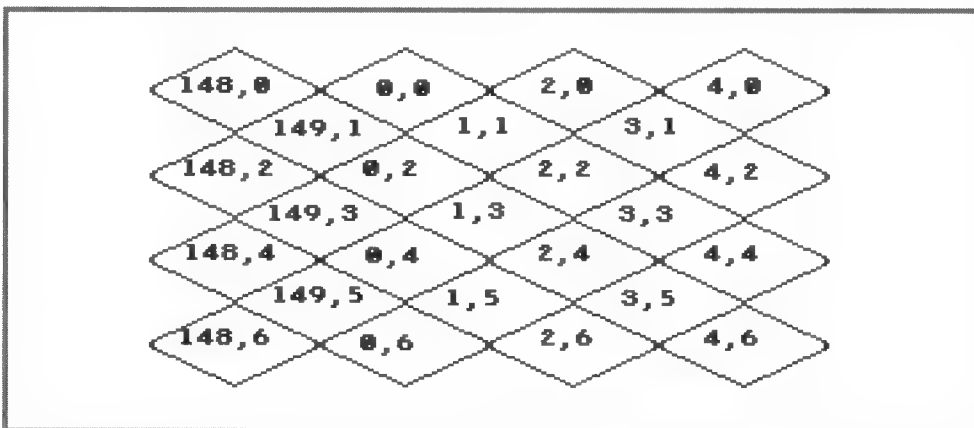
Figure 2-6: The Map Editor grid



If you generate a small map, which is supposed to be 40 x 50 squares, it appears in the editor to have squares numbered from (0,0)—in the upper left corner—through (39,49)—lower right. However, there are two “columns” for each x coordinate—rather than counting 1, 2, 3 ..., they count 1, 1, 2, 2, 3, 3 ... There’s one column each for the even y coordinates and the odd y coordinates. Thus, there are actually 80 columns of squares. That’s where the game handles things differently.

Figure 2-7 shows the numbering scheme used once a game begins. Note that, rather than having two columns for each x coordinate, we now have only one. The map has the same number of squares, and all of them in the same places, but they’re numbered differently (in the case of a small map, (0,0) through (79,49)). The map that is called “40 x 50” in the Map Editor shows its true, 80 by 50 nature after you’ve loaded it into the game.

Figure 2-7: The game map grid



The numbering system recognized by events that you program is the *in-game* system.
If you program location coordinates based on the Map Editor numbering,
nothing happens where you want it to.

SP

Tables 2-2 and 2-3 present lists of a few reference lines and points that I've found useful when building scenario maps.

Table 2-2. Map Editor Reference Points for Standard Maps

SIZE	DIMENSIONS	EQUATOR	MERIDIAN	SQUARE NEAREST CENTER
Small	40 x 50	24/25	20 Even	20,24
Normal	50 x 80	39/40	25 Even	25,40
Large	75 x 120	59/60	37 (Middle)	37,60

Table 2-3. In-Game Reference Points for Standard Maps

SIZE	DIMENSIONS	EQUATOR	MERIDIAN	SQUARE NEAREST CENTER
Small	80 x 50	24/25	40	40,24
Normal	100 x 80	39/40	50	50,40
Large	150 x 120	59/60	74	74,60

The size you decide to set for your map might depend on the map you're planning. For example, if you have a map of Greenland you want to emulate, and that map spans 50 degrees of latitude (north to south) marked in 10-degree increments, it would be convenient to have *y* coordinates that are easily divisible into 50 units. Thus, you would want to build an *x* by 50 or *x* by 100 map, but because of the numbering difference, you would enter the size as *x* by 25 or *x* by 50 (or *x* by 75 for an *x* by 150 map).

Map-Making

The most efficient way to make a new map is different for every individual. If the method described here doesn't work for you, use whatever does work. Here's how we generally make maps at MicroProse:

- 1 Set the size of the world.
- 2 Draw in the coastlines, then fix them in place.
- 3 Fill in the detailed interiors (including rivers and, sometimes, starting locations and resources).

Let's get into the details.

Set the Map Size

When you begin making your map, you should have at the very least a general idea of what you want to end up with—the physical scale of your scenario. Without a pretty firm concept of the size of the world you need, you can't even begin. After all, when you choose to start a new map in the Map Editor, the first thing you must do is set the size of your world.

Draw the Coastlines

Once you've set the size of your world, the toughest part begins—continents and their coastlines. There are two schools of thought on how to build your land masses. Both focus first on determining the coastlines, but in different ways. Mick Uhl, for example, begins by laying down a detailed coast on an ocean background, traces the coastlines, then fills in the middles with a larger brush. Mick traces the coastlines first, because “placing coastlines is the most difficult element of map design for me. Once I have the coastlines in, I don't have much trouble orienting the other features that I want.”

The other method is the one I use—I work from a rough shape and get the details second. With one of the larger brushes (5 x 5 or City Radius), I draw rough, lumpy versions of the continents I want. When I'm happy with the general shapes and sizes, I go back with the single square brush to trim the edges down and put in the details of the coastlines. This method saves me from having to carefully fill in the middles without disturbing my already finished coastlines.

Mick Uhl on Simulating Real Geography

Whichever method works for you, the important lesson is this: fix your coastlines first. It doesn't even matter what terrain type you draw with. After all, you can change all that later. I usually use Grassland or Desert, but it saves a little effort if you use whatever type you think will be most common in your final map. Once your continents are built, activate the Coastline Protect feature on the Tools menu. Now you cannot accidentally modify what you've already done.

Fill in the Interiors

When the coastlines are fixed and locked, do the interiors. It doesn't matter in what order you place types of terrain. Generally, it's easier to go over the whole map and do all of one type at once—all the mountains, for example—but if that's not convenient for your map, do it some other way.

Here are a few bits of geographic theory to keep in mind when attempting to build “realistic” continents:

- ▼ With few exceptions, rivers run downhill, toward the ocean, and toward the Equator.
- ▼ Mountains tend to come in ranges, and they're often bordered by foothills and forested areas. Volcanic mountains are the exception.
- ▼ Inland seas generally are shallower than oceans and, therefore, have more islands.
- ▼ Jungles and Swamps near the poles are as rare as Tundra and Glaciers in the tropics.
- ▼ Due to the prevailing westerly winds on Earth, areas immediately to the west of mountains tend to be wetter (Forest, Grassland, or Jungle), while the areas to the east are drier (Plains or even Desert).

First, I try to find a map of the proper scale. What I mean by “proper scale” is, I locate the coastlines and river systems for a new map by orienting a tracing of their positions (that I made from a map found in an atlas or book) over graph paper. I generally divide each square of the graph paper into four quadrants and match each quadrant to a tile of the game map.

If the map tracing does not properly fit the dimensions of my game map, I either have to reduce or increase the size of the tracing (on a copy machine). If it fits correctly without any need of adjustment, then I consider it in proper scale.

If I find one, which has always been the case so far, I trace the outline onto graph paper, match the graph paper grid to the game map grid, and basically trace. I do not trace on the monitor. I line off the dimensions of the map on the graph paper using the scale of one graph quadrant (equivalent to one quarter of a square), equal to one map tile. I often trace the major river systems as well.

I generally trace the coastlines first, then I put in the river systems. I next place the major Mountain and Hill ranges, then, finally, put in the other terrain. Unusual terrain such as Swamps, Jungles, Tundra, and Glaciers I can put into historical positions, as I usually have a reference map that shows their locations. I also try to put Plains, Forests, and Grasslands into the historic locations, but when they cover large expanses, I randomly intersperse them with other terrain to break the monotony.

Placing Rivers

There are two further rules to remember when you're putting in the rivers:

- 1 Rivers are not an independent type of terrain, they are an addition laid on top of the existing terrain. So, if you put a River on a Desert square, you end up with a River running through the Desert. (Note also that sometimes the terrain underneath ends up looking almost nothing like it did before the river arrived.)
- 2 Once you leave the Map Editor, the rivers are all fixed and cannot be changed. Once you've loaded the map into a game, you can change all of the terrain types—but you can't change the course of a river. So make sure the rivers are exactly the way you want them before you leave the Map Editor.

NOTE

Keep in mind that for the purposes of unit movement, rivers are permanent roads that are available right from the beginning of the game.

Placing Starting Locations

This part is totally optional. If you already know where you want certain civilizations to start the scenario, you can set default starting locations in the Map Editor (see Figure 2-8).

Figure 2-8:
You must assign a
tribe to each
starting location.



Later, when you're building the scenario, you can put cities anywhere you want to, so this might seem sort of pointless. Sometimes, though, the time you spend working on the map gives you a good understanding of how your geography is reflected in your map. As you work on other parts of the scenario, spending time and mental focus on other topics, that geographical understanding can fade. Leaving yourself reminders of where certain cities ought to be isn't a bad idea.

Placing Resources

Scenario designers have very little control over the placement of the villages and other special resources. There are several placement patterns to choose from, but you can't place them exactly where you want them.

Selecting Set Resource Seed from the Tools menu enables you to enter any whole number you want, but it's a bit misleading. There are really only 64 possibilities—and just 63 patterns. For numbers above 64, the patterns start repeating themselves. So, for example, if you enter seed 65, you get seed 1, 66 is identical to 2, and so on.

Seed 1 (and 65, and 129, and so on) is “random.” That doesn't mean that you get a truly random seeding, though. It means that you're letting the computer choose one of the other 63 patterns at random.

What use is this? Sometimes, you want your selected city sites to have a little extra, resource-wise; or you might want to make sure that a certain city does not have more than you've already supplied. Villages, especially (most notably on Deity level), can make the difference between the lucky civilization that gets a boost early in the game and everyone else.

Finding the right seed for your scenario is a process of trial and error. Experiment with the 63 seed patterns until you find one that fits your needs. Keep in mind, too, that when you're setting up the scenario situation (for more details, see Step 7: “Scenario” Means “Situation”), you can't change the resource seed, but you *can* eliminate all the villages.

Customizing Terrain

Now for some extra fun. There's absolutely no reason to limit yourself to working with the terrain types (and special resources) that *Civilization II* provides. If the traditional output from the various types of terrain doesn't suit your needs, you can change it all. If your scenario seems to call for unusual or even unearthly terrain, that's no problem, either. *Civilization II* enables you to invent your own terrain types.

If you are making your changes in a complete duplicate copy of *Civilization II* (not using a scenario folder), your changes to the terrain will be reflected in the Map Editor in that directory. Otherwise, if you'd like to see your terrain when building maps, you must copy your edited *rules.txt*, *terrain1.gif*, and *terrain2.gif* into the base folder. (Make backup copies of the original files first!) That's how I made the Mars map (see Figure 2-9).

CIV II

Figure 2-9: Custom terrain types in the Map Editor



You might choose to do all your terrain modification before you begin building your map (as I did when building the Mars map). That way, you can see what you're really working with. If your modifications are fairly minor, though, it probably makes no difference during the map-making process. You can do both simultaneously, too. In scenario design, when you get an idea, run with it!

Characteristics

In the long run, the way your new or improved terrain types function is more important than the way they look. Balance is vital. You've got to give players enough of everything, resource-wise, but not too much of anything.

If you have *Fantastic Worlds*, you can modify the characteristics of every terrain type and natural resource using the Terrain Editor on the Editors menu. While this method is more convenient and probably easier for most folks, it is only available after you have created and loaded your map.

Understanding the terrain type definitions is essential. The types of terrain in your scenario world are defined in *rules.txt*. Search through that file until you find this line:

@TERRAIN

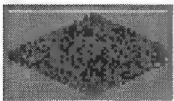
This is the header marking the beginning of the section containing the definitions of every terrain type and special natural resource. Each of the lines of text following it—up to the next header (**@GOVERNMENTS**) determines the characteristics of a single terrain type. (As far as the program is concerned, special resources are sub-types.) Editing these definitions, you can change nearly everything about a type of terrain.

All the lines that start with the @ symbol are header lines. Do not *ever* change any of the header lines. If you do, *Civilization II* will probably crash. Also, you should never add lines to or delete lines from *rules.txt*—not even empty lines. In some cases, the program counts lines to find data, and if you remove or add something, you can only cause trouble.

CAUTION

Let's take one of these definition lines apart. The Grassland type is defined as:

```
Grassland 1 1 2 2 1 0 yes 1 5 2 For 0 1 0 0 Hil 1 Grs
```



Each part of the line is a data field, and each field except the last one must end with a comma. Table 2-4 covers the meanings of all the fields.

Table 2-4. Terrain Data Fields for Grassland Example

VALUE	FIELD NAME	EFFECT
Grassland	Name	This is the name of the terrain type. It shows up in any on-screen text about the terrain (mostly in the Status box). Like the manual states, names shouldn't be more than 15 characters, because they won't fit in all the display boxes and lists. Also, you should use only letters, numbers, and spaces in names; other characters might not be displayed and can occasionally cause program errors.
1	Move	This is the number of movement points a unit spends to move onto a square of this type of terrain. Because of the exception that every mobile unit can move at least one square per turn, any number higher than the fastest unit's movement allowance is pointless. This field can have a value of zero.
2	Defense	The defense rating determines—in 50 percent increments—what percentage of a unit's Defense Factor is applied in combat when that unit is standing on this type of terrain. If this is 1, for example, the unit defends at 50 percent (half). A defense of 2 is normal (100 percent), and anything over that (3 or higher) adds to the unit's defense. Fortresses can increase this, so be wary of setting this too high. <i>This field should never have a value of zero.</i>
2	Food	This one is simply how much food a square of this terrain produces. This is the base number, which can be changed by many things, including irrigation and city improvements.
1	Shields	This is how many production shields a square of this terrain produces. Like Food, this is the base number, which can be changed by the player with things like roads and city improvements.



VALUE	FIELD NAME	EFFECT
0 𐌹	Trade	This field sets the amount of trade (arrows) a square of this terrain produces. As with Food and Shields, this is the base number.
yes 𐌹	Irrigate	You can control what types of terrain civilizations can irrigate and the results of that irrigation. <i>Yes</i> in this field means that the terrain is irrigable and the Irr Bonus applies; <i>No</i> means that this terrain is not irrigable at all. If you put the Abbreviation of one of the other terrain types here (the <i>exact</i> abbreviation), then irrigating this type of terrain will change it to that other type.
1 𐌹	Irr Bonus	When this type of terrain is irrigated, it produces this much <i>more</i> food. If you set Irrigate to anything other than <i>Yes</i> , then this field should be zero.
5 𐌹	Irr Turns	This determines how many turns it takes a Settler to irrigate a terrain square of this type. An Engineer does it in half this number, rounded up to the nearest full turn. If you set Irrigate to <i>No</i> , then this field should be zero; otherwise, it should not be less than 1.
2 𐌹	Irr Govt	The computer-controlled civilizations don't bother with terrain improvements until it's cost-effective. This field tells them when that is. An AI civilization won't irrigate squares in its city radii until that empire discovers (successfully researches) the level of government specified here or a higher one (see the sidebar The Levels of Government in <i>Civ II</i> .) If this field is zero, the AI will <i>never</i> irrigate.

Table 2-4. Terrain Data Fields for Grassland Example (continued)

VALUE	FIELD NAME	EFFECT
For	Mine	Like irrigation, you can also control what types of terrain can be mined and the results of mining. <i>Yes</i> in this field means that mining is possible and the Mine Bonus applies; <i>No</i> means that this terrain can't be mined. If you put the Abbreviation of one of the other terrain types here (the <i>exact</i> abbreviation), then mining this type of terrain will change it to that other type.
0	Mine Bonus	When this type of terrain is mined, it produces this many <i>more</i> production shields. If you set Mine to anything other than <i>Yes</i> , then this field should be zero.
10	Mine Turns	This determines how many turns it takes a Settler to mine a terrain square of this type. An Engineer does it in half this number, rounded up to the nearest full turn. If you set Mine to <i>No</i> , then this field should be zero; otherwise, it should not be less than 1.
0	Mine Govt	This field is like Irr Govt; an AI civilization won't mine squares in its city radii until that empire discovers the level of government specified here or a higher one. (See The Levels of Government Sidebar for the levels.) If this field is zero, the AI will <i>never</i> mine.
Hil	Transform	This is the result of an Engineer unit performing Transform orders on a square of this terrain type. If you put the Abbreviation of one of the other terrain types here (the <i>exact</i> abbreviation), then Engineers can transform this type of terrain to that other type. How long the transformation will take is determined elsewhere (see Step 6: The Foundations of Civilization). If the value in this field is <i>No</i> , Engineers cannot transform this terrain. (Note that “Engineer” means any unit in unit slot 2 that has a Role of 5 (for more details, see Step 3: Design Units).



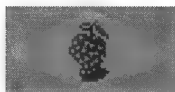
VALUE	FIELD NAME	EFFECT
GrS	Abbreviation	This is the official abbreviation for this terrain type. <i>Do not change this.</i> It doesn't matter what the name of the type is; this is always the abbreviation for the type defined on this line.

Engineers do *not* complete terrain improvements twice as fast as Settlers because they have double the movement allowance (Settlers = 1, Engineers = 2). The Engineer's speed is a function of the unit slot it's in. (For more details, see Step 3: Design Units.)

TIP

The special natural resource definition lines are truncated versions of the full terrain type definition. These have only the first six fields: Name, Move, Defense, Food, Shields, and Trade. (Notice the comma after the last field—it's necessary.)

Wine 2 4 1 0 4 ,



When the resource appears on a square, these numbers supersede the values for those fields of the terrain type associated with the resource. The rest of the fields are not affected.

The Levels of Government in *Civilization II*

These are the "levels" of government in *Civilization II*:

- 1 = Despotism
- 2 = Monarchy
- 3 = Communism
- 4 = Fundamentalism
- 5 = Republic
- 6 = Democracy

The resource entries are in order. Specifically, the resources are listed in the same sequence as the terrain types they correspond to. If you look at Table 2-5, you'll see that everything is listed in the same order as it appears in *rules.txt*; I've just put it into columns. (There are two possible resources for each terrain type.) If irrigation, mining, or a transformation changes the terrain type of a square, any special natural resource on that square changes to match the resource of the same number (Resource 1 or Resource 2) for the new terrain type.

Fantastic Worlds adds *ChangeTerrain* to the actions you can include in an event. (If you don't know what events are, see the scenario pack manuals and Step 8: Schedule Your Events.) In the parameters for that action, you must specify a terrain type by number.

Table 2-5 lists the numbers for you. Special resources change along with the underlying terrain.

FANTASTIC
WORLDS



Table 2-5. Terrain Types and their Resources

#	TERRAIN	RESOURCE 1	RESOURCE 2
0	Desert	Oasis	Desert Oil
1	Plains	Buffalo	Wheat
2	Grassland	Grassland	Grassland
3	Forest	Pheasant	Silk
4	Hills	Coal	Wine
5	Mountains	Gold	Iron
6	Tundra	Game	Furs
7	Glacier	Ivory	Glacier Oil
8	Swamp	Peat	Spice
9	Jungle	Gems	Fruit
10	Ocean	Fish	Whales

So now that you understand the characteristics of terrain types that you can control, what good is it? Well, if reading through the descriptions didn't give you ideas, maybe a few examples will spark your imagination:

- ▼ **Relics** could be a special resource that adds Trade (science, income, and luxuries) or Shields—in a magic-oriented game, ancient relics could be a source of power—but lessens Food because some of the potential farmland is given over to digging for relics.
- ▼ There are already Forests, but you might want a **Haunted Forest**, too. Food, Shields, and Trade would be very low—maybe even zero. Cities on the edge of these woods wouldn't grow too quickly; maybe sometimes the citizens mysteriously disappear. Movement through this terrain would be extremely slow, but defense could be high (because no one wants to go in there after you) or low, so that the monsters in the forest have the advantage.
- ▼ Perhaps, long ago, a mysterious race planted the seeds of change. One type of land, when irrigated, changes into a sea of bizarre alien **Greeble** plants (or creatures or ooze or anything you like) as the ancient seeds awaken and grow. The characteristics of this terrain could be anything.

- ▼ Mining isn't always careful. You could set it up so that mining changed certain types of terrain into a **Wasteland**. Shields would be high, but the land would produce almost nothing else.

Good planning is only half the work. After your terrain types are complete, play a few games with the new characteristics. That way, you'll find out for yourself whether you have created a balanced set of land types. If cities grow too fast and too easily, you need to lower the numbers of what terrain types produce. If cities grow too slowly or too much terrain improvement is necessary, you might need to raise the numbers. Both of those considerations are dependent on your design, though. You might *want* things to be extra easy or rather tough. For example, in the Age of Reptiles scenario, I fully intended city growth to be slow and a little difficult, so the terrain statistics are a bit low.

Appearance

What good are new types of terrain if they look the same as the old ones? Well, again, that depends on the design of your scenario. Historical setups are much less likely to require a new look, so the original *Civ II* terrain might do just fine. If you want a new appearance for your terrain types, though, it's not quite as easy as changing the characteristics; you need to be able to edit GIF files.

If you have *Fantastic Worlds*, you can modify the looks of every terrain type, natural resource, and all the miscellaneous stuff using the Terrain Editor on the Editors menu. While this method is more convenient and easier than a GIF editor for most folks, it is only available *after* you have created and loaded your map. (For more details on GIF editors, see Step 1: Getting Started.)



The remainder of this section assumes that you have a GIF editor or *Fantastic Worlds*.

All the terrain graphics, plus some of the miscellany that can show up on terrain—roads, railroads, mining, resources, irrigation, and the like—are contained in two files: *terrain1.gif* (see Figure 2-10) and *terrain2.gif* (see Figure 2-11). Table 2-6 is a reference list of what is where.

Figure 2-10: Terrain1.gif

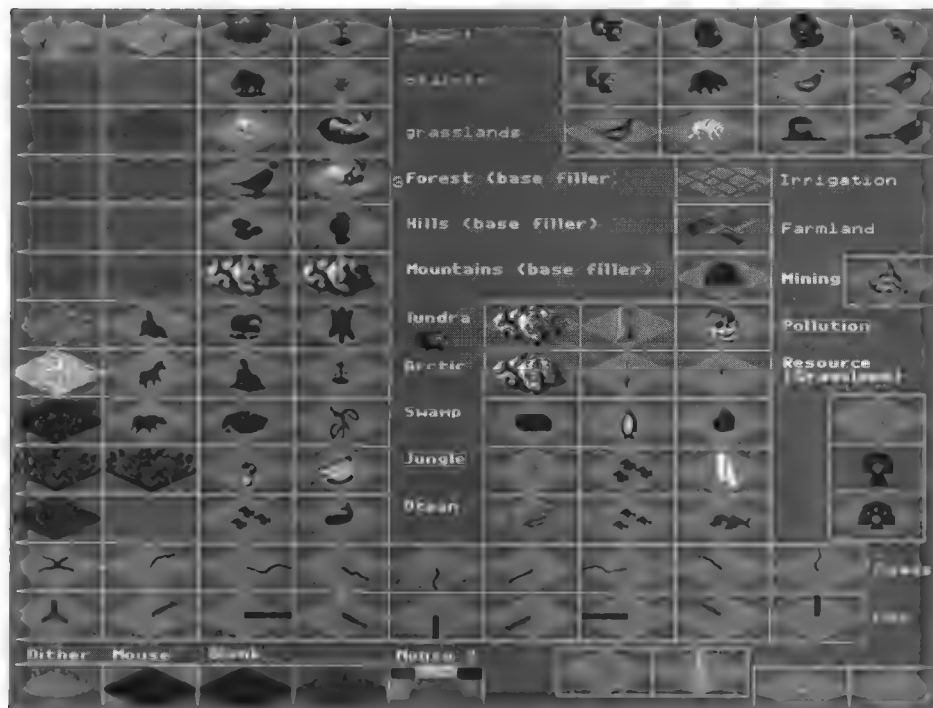


Figure 11: Terrain2.gif

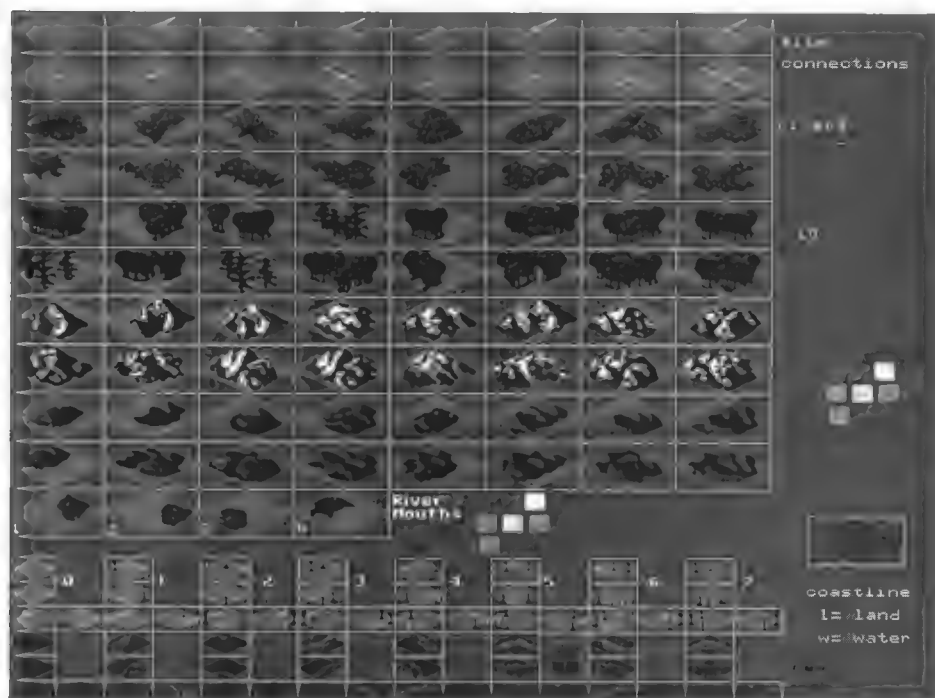


Table 2-6. Terrain Graphics: What's Where

GRAPHIC	FILE
Buffalo	Terrain1
Coal	Terrain1
Coastlines	Terrain2
Desert	Terrain1
Desert Oil	Terrain1
Farmland	Terrain1
Fish	Terrain1
Forest base	Terrain1
Forest trees	Terrain2
Fruit	Terrain1
Furs	Terrain1
Game	Terrain1
Gems	Terrain1
Glacier	Terrain1
Glacier Oil	Terrain1
Gold	Terrain1
Grassland	Terrain1
Grassland Shield	Terrain1
Hills	Terrain2
Hills base	Terrain1
Iron	Terrain1
Irrigation	Terrain1
Ivory	Terrain1
Jungle	Terrain1
Mining	Terrain1
Mountain base	Terrain1
Mountains	Terrain2
Musk Ox	Terrain1

Table 2-6. Terrain Graphics: What's Where (continued)

GRAPHIC	FILE
Oasis	Terrain1
Ocean	Terrain1
Peat	Terrain1
Plains	Terrain1
Pollution	Terrain1
Railroads	Terrain1
Rivers	Terrain2
Roads	Terrain1
Silk	Terrain1
Spice	Terrain1
Swamp	Terrain1
Tundra	Terrain1
Village	Terrain1
Whales	Terrain1
Wheat	Terrain1
Wine	Terrain1

NOTE

If you have *Fantastic Worlds*, take a look at page 46 in the manual. On the screen shot, there's a button marked Edit Misc. Read the text, though, and you'll find no mention of it. (*Mea culpa*. I wrote the damn thing.) That button enables you to change the appearance of Roads, Railroads, Mining, Irrigation, Farmland, Villages (Goody Huts), Pollution, and the Grassland special resource (Grassland Shield).

Once you understand the way these files are laid out—what's what, how you can change the game, and what you shouldn't mess with—you'll be able to do nearly anything with the looks of the *Civ II* terrain.

The Palette and Green Lines

Files in GIF format are one example of what are called *paletted* graphics files. The palette (sometimes called a “color table”) is a collection of colors defined for use in the file, and no other colors appear in that file. Most graphics programs include a way to display the palette, usually as a box full of colors you can pick from. All of the files for *Civilization II* use the same palette, and you should never change the palette in any of the files—ugly graphic problems result, and the game could crash, too.

Here’s an example of how fruit looks in the file



and what it looks like in the game



The *Civ II* palette contains some colors that are recognized by the program as “not to be displayed” colors. These are mostly “transparent” colors that we can use in the files to help us position graphic elements correctly. For example, the special resource icons all sit on a purplish background in the shape of a normal terrain square (also called a “tile”). That purplish diamond sits on a grey background. Both the purplish color and that exact shade of grey are transparent colors. When the resource icon is displayed on its corresponding terrain, you can still see the terrain underneath—in every place that the purplish color and the grey appear in the graphic file. The purple tile gives you a way to know where on the terrain tile the icon will appear.

Some colors in the palette are very similar to the transparent colors, especially the transparent shade of grey. If you get them mixed up, your game will look pretty bad, but probably won’t crash. Just re-edit the file and replace the wrong color with the right one, and the problem is solved.

CAUTION



Another color that you must be careful about is the bright green used to draw all the boxes around the icons. Those green boxes are used by the program as reference to tell where the icons are. Every terrain icon is in a box of exactly the right size and shape, and you should never mess with the green lines. Work inside the lines.

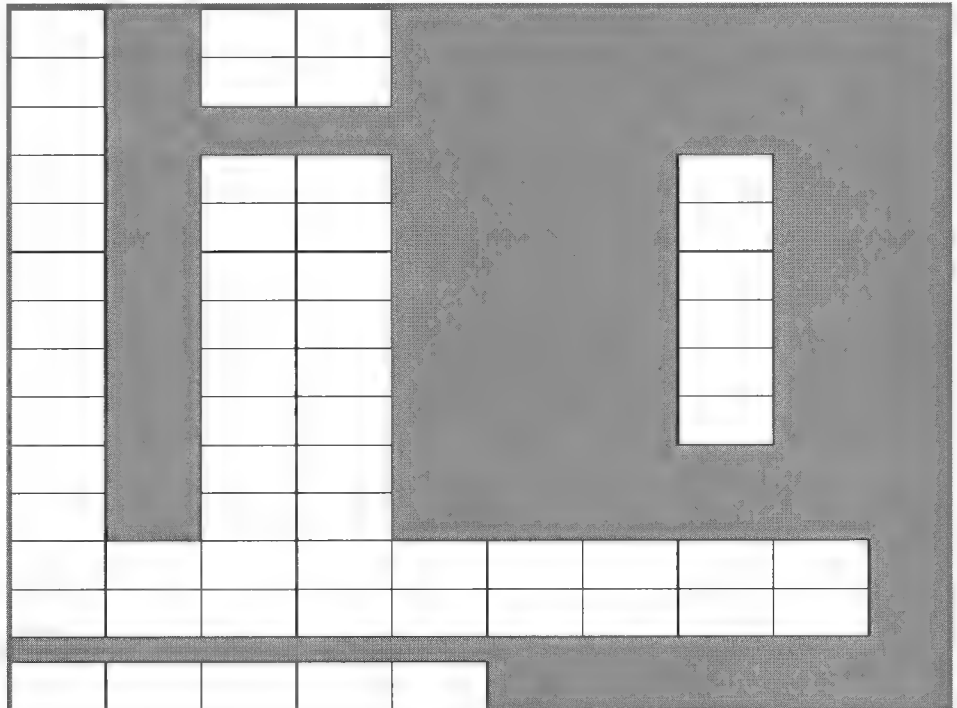
CAUTION

Never move or erase any of the green lines. Don't add new ones, either.

Active Areas

If you look in the files, you'll notice several graphics and other markings that aren't listed in Table 2-6. The MicroProse artists left lots of tools and unused icons in the files, plus there are some bits that the program uses. Anything that does not appear in the game is in an *inactive area* of the GIF file. Anything outside of the *active areas* is either ignored by the game program or is something you should never touch, so there's no point in changing any of it. Figures 2-12 and 2-13 show active areas of *terrain1.gif* and *terrain2.gif*, respectively.

Figure 2-12:
Active areas of
terrain1.gif



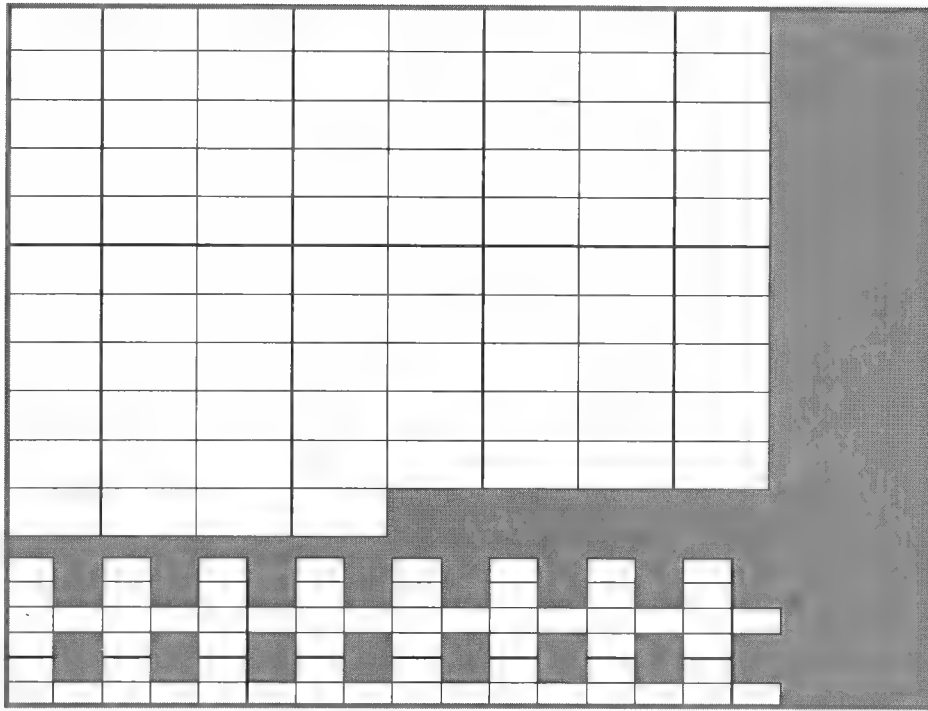


Figure 2-13:
Active areas of
terrain2.gif

For example, in *terrain1.gif*, there are two alternate Silk icons to the right of the two Grassland tiles. Those aren't used in the game, so they're inactive. The Silk icon just below them (next to the Forest base tiles) is active. If you copied one of the inactive icons and pasted it over the active one, the icon you pasted would become active. It's the placement, not the icon itself, that matters.

Base Filler Tiles

Take a good look at *terrain1.gif* (the file or Figure 2-10) and you'll see that the tiles representing Forest, Hills, and Mountains are exactly the same. They're not the same in the game, so what's the scoop? These tiles are only half of the terrain.

Remember the note about how River tiles are placed on top of other terrain to make terrain-with-a-river? Well, the game displays Forest, Hills, and Mountain terrain in a similar way. The tiles in *terrain1.gif* are the *base filler* tiles for those three terrain types. The Mountain, Hills, and Forest tiles in *terrain2.gif* do not cover the entire terrain square; there's some purple around them and in between. Wherever that purple appears, the base filler shows through.

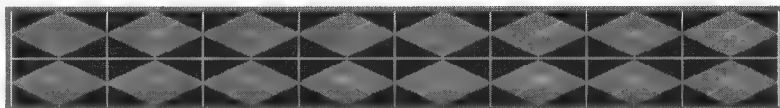
Multiple Tiles

Not all of the terrain types in *Civilization II* are defined by just a single graphic tile. In fact, only eight of them are. Plains, Ocean, Tundra, Glacier, Grassland, Jungle, Desert, and Swamp are the simple ones; change one tile, and you've changed the look of that terrain type.

NOTE If Desert, Jungle, and Grassland are defined by one tile, why does *terrain1.gif* include two tiles for each of them? The second tile is an alternate that MicroProse decided not to use. The alternates for Forest, Hills, and Mountains base filler tiles are inactive, too. You can ignore these, or use the spaces (like I do) as convenient boxes in which to make changes before copying them over into an active area.

The others—Mountains, Forest, and Hills—are much more difficult to change, because each one is defined by no less than sixteen tiles! The display of Rivers works the same way and uses the same pattern. What pattern? Take a look at the top of *terrain2.gif* (the file or Figure 2-11). The uppermost two rows of tiles make up the Tile Connection Plan (shown by itself in Figure 2-14). This plan shows you how the River, Forest, Mountain, and Hill tiles are placed in the game.

Figure 2-14: The Tile Connection Plan



A couple of examples are definitely in order. Tile 1, the first one in the first row, has just a dot in it. This represents the case of a single tile with no tiles of the same type adjacent to it. (“Adjacent” in this case means touching it along any of the four edges; diagonals don’t count.) The corresponding tile for each of the terrain types—the tile in the leftmost upper position—is the one displayed in the game for a lone square of that terrain type—a single mountain, for instance, or a lone patch of forest.

Tile 8, the last one in the upper row, has lines radiating from the center dot to its northeast, southeast, and southwest edges. That means that this tile is bordered on those three sides by terrain of the same type. The corresponding tiles for Forest, Mountain, Hills, and River—the last tile in the upper row of each—is the one displayed in a terrain square bordered in the same pattern by squares of the same type of terrain.

Why go through all this trouble? Well, take Rivers as an example. If every River tile looked exactly the same, the map would look terrible—all the rivers running in the same direction, not matched up or anything (see Figure 2-15). The sixteen tiles link up in a predetermined pattern to make much more realistic and enjoyable maps.

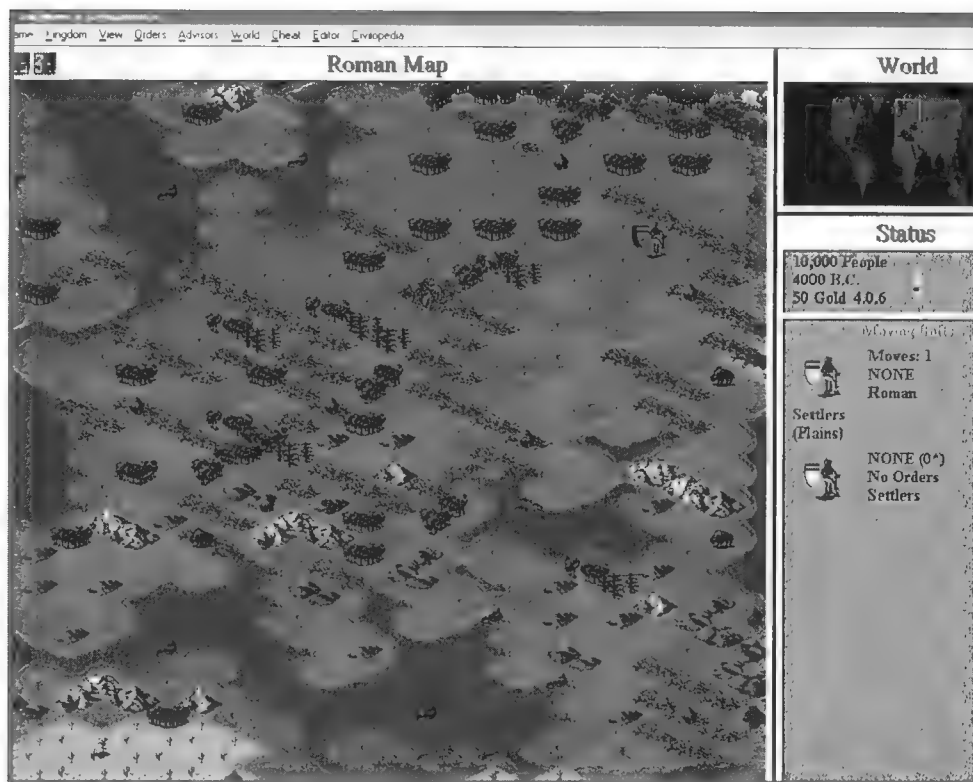


Figure 2-15: How would you like it if rivers looked like this?

The coastlines that separate Ocean squares from all others work in a similar fashion. At the bottom of *terrain2.gif* are all the mini-tiles used to display coastlines. Immediately above them are the tiles that make up the Coastline Connection Plan. This works just like the Tile Connection Plan, except that instead of radiating lines, the edges and corners are marked with W for water and L for land—telling you what that portion of the tile connects to.

The problem comes when you start changing these tiles—terrain, river, or coastline. The existing tiles are very carefully drawn so that they match up smoothly in the game. Getting your tiles to match up so well can take a lot of time and effort. It's worth it when your map comes out perfect, but that amount of trouble is not to everyone's taste.

Do Not Touch

Here's a general rule that I apply to all the *Civ II* files: If you don't know what it is and how it works, don't touch it.

When it comes to the terrain files, there are only three parts (other than those already mentioned, like the green lines) that neither you nor I nor anyone else should change:

- ▼ At the bottom left of *terrain1.gif*, there are five tiles set off by themselves. Four are labeled “Dither,” “Mouse,” “Blank,” and “Mouse 2.” The *Civ II* program uses these for various purposes, so they’re off limits.
- ▼ At the top of *terrain2.gif*, as I described a little earlier, is the Tile Connection Plan. Editing it is a bad idea.
- ▼ Also described earlier is the Coastline Connection Plan, just above the coastline tiles. Hands off.

Avoiding Map Problems

When you have your map open in the Map Editor, you can use the Analyze Map option on the Map menu to check it out. Doesn’t a clean bill of health from the analyzer ensure that your map is a legal *Civ II* map that will work just fine? Well, no, it doesn’t. The map analysis tool checks for two things—and only two things:

- 1 Does the map have enough land squares? In my series of unofficial, incomplete, and totally unscientific tests, I determined that on a medium-sized map, “enough” land means at least eight city radii (see Figure 2-16), plus or minus a square or two.
- 2 Are there at least seven possible sites for cities? The analyzer makes sure that there are at least seven or eight Grassland squares, and leaves it at that.



Figure 2-16: Try it.
It's a legal map.

These two criteria make a “legal” map for *Civilization II*, but they absolutely do *not* make a map that will not cause the game to crash. That’s because there’s another criterion that the analysis tool doesn’t look for: Are there enough Ocean squares?

Where’s the Beach?

The lack of Ocean squares is known to cause two serious problems, and there might be others that have, as yet, gone undiscovered. First and most important, if there are not at least twenty or so Ocean squares on the map, the game locks up—simply stops working with no notification that there’s a problem. Sometimes this happens right at the start of the game, but more often you’re thirty to a hundred turns into the game before things freeze up, which is more than a little annoying.

Not Enough Ocean in the Mars Scenario

Those of you who have played *Fantastic Worlds* know that the map for the Mars Now! scenario is a fairly accurate representation of the planet Mars, except that there are a number of Ocean squares near the polar caps. The manual says that this is because early colonists began melting the ice caps, but that's just an excuse.

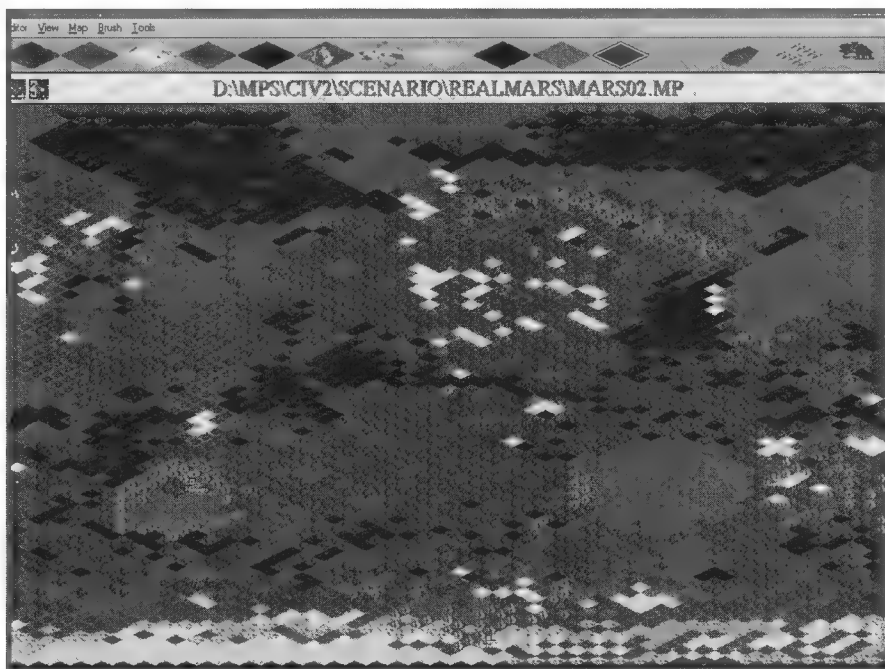
According to the original design, there were to be only enough Ocean squares to support the capital cities near the poles. If the civilizations wanted water, they would be forced to mine the Icecap terrain (Glacier renamed) to create new Watersheds (Ocean renamed). That didn't work out too well.

Without exception, every time anyone tried to play the Mars scenario, it locked up between turns 29 and 32 (most often right between turns 31 and 32). As it turned out, there simply were not enough Ocean squares on Mars for the game to function properly. Thus, we were forced to introduce the idea that the colonists had done a little early ice melting.

Even if you have enough Ocean to prevent the game from having a seizure, there's another little problem—Goto orders don't work right. Whenever you give a unit a Goto order, you set a destination and allow the game to figure out the most efficient way to get there. (In fact, the program performs poorly in that respect when there are railroads involved, but I digress.) One of the criteria that the Goto function looks for when calculating the best route is the locations of the nearest coastlines. If the coast is too far away, the calculations go awry. Therefore, if the coastlines are too few or the continents too large on your map, units given Goto orders will go the long way, wander around aimlessly, possibly never reach their destination at all, and sometimes just stand still in frustra-

tion (see Figure 2-17). For short routes, the problem is almost nonexistent, but the farther away the destination, the more chaotic the unit's path becomes.

Figure 2-17: Goto orders with faraway destinations don't work well here.



This wouldn't be a big problem for a human player, but for the computer-controlled civilizations, *every time* a unit moves, it is the equivalent of a Goto order. You can imagine the results. (Or you can go play the Mars Now! scenario and see it for yourself; that scenario has few enough Ocean squares that the AI units sometimes get lost.)

The obvious solution to these problems is to put plenty of Ocean squares in your map, but in some cases, that's not appropriate. Let's say your scenario takes place on a desert world, in an underground kingdom, or some other place where oceans would be out of place. Take the X-COM: Assault scenario as an example; that scenario takes place on a moon of Mars that has no liquid water at all, much less oceans. What did Bob do? He replaced Ocean with another form of terrain—Sulphura (see Figure 2-18).

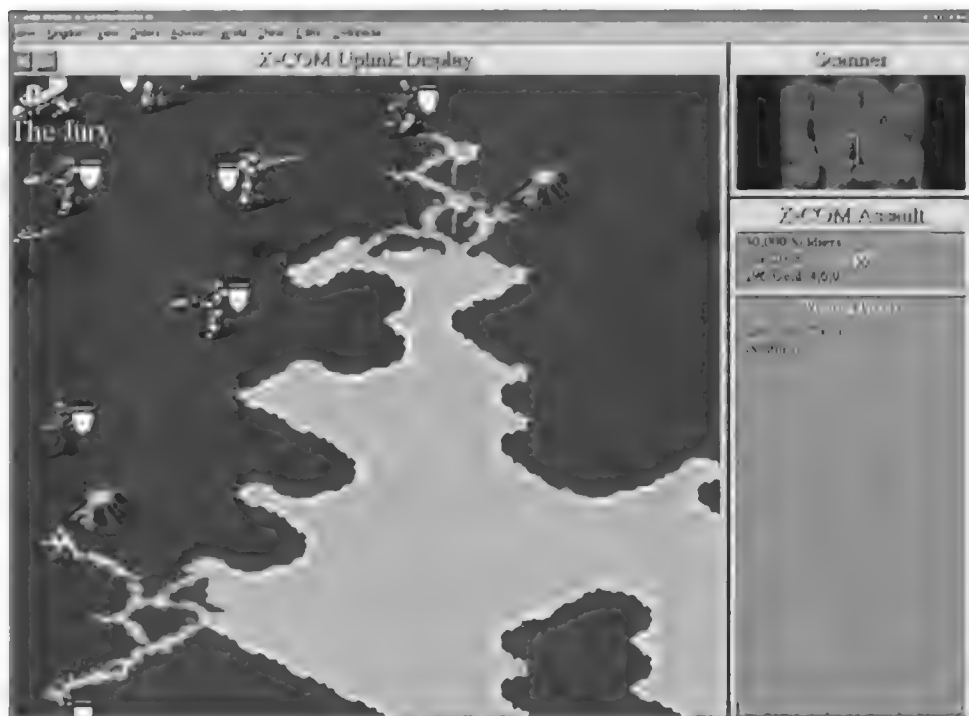


Figure 2-18:
Sulphura is really
Ocean.

We'll get to how you can replace Ocean yourself toward the end of the chapter. The key to solving the inappropriate Ocean problem is this: Whatever type of terrain is defined in *rules.txt* in the place of Ocean will be Ocean as far as the game program is concerned. That means that irrigation is possible next to it, naval units can travel on it, air units can fly over it, but ground units and cities are forbidden there. As long as there is enough of that type of terrain in your map, you will have no Goto problems.

Confusing the AI

The other noticeable problem you can cause by careless map design is less troublesome, but can still ruin an otherwise fun scenario. If you do not supply enough decent city sites, the computer-controlled civilizations have trouble acting rational.

Human players can figure out when a city site needs to be prepared ahead of time—for example, in resource-poor areas, a little pre-irrigation is sometimes necessary to provide an ample food supply for a new city. Engineers can do wonders to make an otherwise useless area into a decent city site. However, the AI doesn't think in those terms; if a site doesn't make the grade without improvements, no city goes there. (It's actually more complicated than that, and the AI will do some things when land is scarce that it would not normally do, but as a general rule, this is as good as the whole truth.)

The end result is that computer-controlled civilizations offer little challenge on maps that contain little if any useful terrain. That's one reason why desert world scenarios are so difficult to do well—and why the terrain in the Mars scenario produces so much more than real Martian terrain is likely to offer.

You could make this situation even worse (were you so inclined) by establishing cities for the computer-controlled civilizations in resource-poor sites. Again, human players have an advantage; the AI almost never uproots a city.

The point is that you should always provide enough good sites for cities and make sure that the preexisting cities you put in place are able to support themselves. If you don't want anyone to build new cities—which is often the case, especially in historical scenarios—this is not the way to do that. Make sure that no Settler-type units exist and none are available for building (for details on how to do *that*, see Step 3: Design Units), and you won't need to worry about anyone establishing cities.

Using Your Map

Prepare yourself. When you fire up *Civilization II* and load your carefully planned, perfect map, the game is going to try to make some changes to it. Why? Well, there's one characteristic that every map must have in order for the game to start up correctly, and if your map doesn't meet the requirement, it's forced on it. There are also a few options you need to be aware of to avoid accidents—choices you make during the game setup supersede those you made while building the map.

- ▼ *Civilization II* doesn't like to start a game using a premade map unless that map has defined polar caps—land all the way across the top and bottom edges (see Figure 2-19). The game doesn't bother to check for this, and if you don't use the Customize Rules option, it just imposes a preemptive fix. No matter what type of terrain you've put along the top and bottom borders of your map, they are replaced with the typical *Civ II* polar caps—a straight line of Glacier one square thick.
- ▼ If you took the time to set a specific (non-random) resource seed for your map, you can lose it during the game setup. A prompt asks, “Do you wish to randomize this world's villages and resource squares?” Answer No if you still like the resource distribution you established; otherwise, resources and villages are reorganized at random, as in any other game.
- ▼ If while building the map you placed starting locations for any civilizations, the game enables you to undo that decision. During the game setup, a prompt asks, “Do you wish to randomize this world's player starting locations?” Answer No if you still like the locations you decided on; otherwise, the civilizations are scattered around the map at random, as in a normal game.
- ▼ When you read Step 7: “Scenario” Means “Situation”, you'll find out why you might choose to use the Customize Rules menu during the game setup process. If you do customize, there's one choice that could affect your map. The Flat World option should reflect whatever shape you set for your map—it's checked for Flat, empty for Round. This is your last chance to change your mind; just make sure you don't switch it by accident.



Figure 2-19: *Civilization II* doesn't like maps without polar caps.

The best way to avoid the enforced polar caps is to simply go through the Customize Rules options; even if you don't use any of the custom rules, you prevent the glaciation. Another good strategy is to expect the polar change and design your map to minimize or completely eliminate any problems it might cause. Unfortunately, that's not always possible. I'll just give you one example. If the play balance of your scenario somehow depends on keeping the civilizations separated for a while, the guaranteed polar route created by the addition of Glacier squares is a problem. Experienced human players know that there's a polar route, and they'll use it.

If you pay attention and select carefully during the game setup, you avoid all but one of these inconveniences. Thankfully, even if you forget to use Customize Rules, there's a way around the polar caps, too. Once you've started the game—you're ready to make your first save file—you can use the Edit Terrain feature on the Cheat menu to get rid of those annoying Glacier squares. They won't come back.

Here's the catch: every time you start a new game using the map, you have another chance to use Customize Rules. If you forget again, you'll need to redo all these little changes. So any time you need to restart the process of building the scenario, whether it's because you had to make major changes to the map or you forgot to keep a save file or whatever, remember Customize Rules or those polar caps are going to creep back in there. (Yet another reason why save files are so gosh darned important.)

TIP

After you've begun building the scenario, you should make whatever little changes to the land become necessary using the Edit Terrain option on the Cheat menu—if it's at all possible. (Some changes are so widespread that it's easier to use the Map Editor.) If you use the Map Editor to change the base map, the only way to get the changes into the scenario is to rebuild the scenario from scratch. This isn't a problem if you've just started, but if you've spent a lot of time setting up a complex situation, it ain't fun. ✂

STEP 3

DESIGN UNITS

BE HONEST—WHEN YOU STARTED THINKING about designing a scenario, what was the first thing you thought about? Cool new units, right? Of course it was. I don't mind admitting that that's how the Age of Reptiles scenario got its start. I wanted to run rampant around the *Civ II* map with a Tyrannosaurus Rex! (I got my wish, too.) This step is all about designing the parts of your scenario that will move, attack, defend, trade, spy, improve, and pillage. You'll get a few guidelines on smart unit design, plus everything you need to know about the unit characteristics and attributes. I'll tell you how to make your own unit icons. Last, but not least, I've included here—for the first time anywhere in print—a guide to how *Civilization II* assigns sounds to units.

General Guidelines

The units you use in your scenarios are likely to be of varied and unpredictable design. If you're contemplating a historical reconstruction, the period and the setting will to a large degree determine your unit designs for you. If you're inventing a world and a situation whole cloth, only the broadest advice will be relevant. I'll leave the creative aspects to you, but I can offer a few guiding suggestions.

- ▼ **Keep the units consistent with the milieu.** This is no less true in fantasy than in historical scenarios. A Nuclear Missile would be just as incongruous in a fairy realm as in the hands of the Hittites. Out of place units can be used to humorous effect, but make sure it's intentional.
- ▼ **Don't make any unit too powerful.** Superunit—it's the first thing most players make when they start creating units. Unbeatable units are fun to play around with for a few minutes, but that's all. In a scenario, they're the kiss of death. Either the player has them, which makes the scenario boring—it's no fun if there's no risk of losing—or the enemy has them, in which case the player hasn't got a chance. If everybody can make the super-unit, the situation's a little better, but you've got to make sure everyone gets them at nearly the same time (like nukes), and there's some sort of balancing penalty or cost attached to building or using the unit (like nukes).
- ▼ **Useless units are pointless.** Every unit you design should have an advantage that tempts the player to build a few. For example: Warriors are cheap and available early, Chariots move quickly, and Catapults have a high attack factor. Some units are always more attractive than others, depending on the player's style, but none of them should be something that no one will ever build.
- ▼ **Match obsolescence with innovation.** When one or more units become obsolete, it's a good idea to have that same advance make at least one comparable new unit available.

When Mick Uhl is designing units for a scenario, he says that the setting is key.

Which kinds of units I choose to include in a scenario, naturally, depends on its theme. I'll try to include at least one, often several, experimental or ground-breaking units in a historical scenario or interesting monsters or war-machines in an ahistorical one. For example, I knew I wanted a Zeppelin in the World War I scenario, and I couldn't imagine a mythical fantasy world without a dragon or an Alien Invasion without an ultra-cool alien space ship.

Mick also suggests that, unless the specific setting or situation of your scenario makes it inappropriate, you should provide every civilization with at least one example of every class of unit. What do I mean by “classes” of units? Glad you asked; here they are:

- ▼ **Defensive Ground Unit** is the city defender. This unit has a good (but not great) Defense Factor, an unimpressive Attack Factor, and a low movement allowance. (Example: Warrior)
- ▼ **Attack Ground Unit** is the explorer/conqueror. A moderate Attack Factor is a must, the Defense Factor is unimportant, and this unit’s movement allowance should be slightly higher than average—at least 2. (Example: Horsemen)
- ▼ **Elite Defensive Ground Unit** is the upgraded version of the Defensive Ground Unit. A higher Defense Factor is a must, and better Hit Points and Firepower are helpful. (Example: Phalanx)
- ▼ **Elite Ground Attack Unit** is the successor to the Attack Ground Unit. This unit should be more mobile (faster) than its predecessor, and may be more powerful. (Example: Armor)
- ▼ **Artillery Unit** is the slow, powerful attacker. This unit should have a low Defense Factor and a very high Attack factor, but be quite limited in terms of speed. (Example: Catapult)
- ▼ **Basic Sea Unit** makes ocean travel possible. This unit should be able to fight *and* carry ground units. (Example: Trireme)
- ▼ **Bomber Unit** is the aerial scout and defense softener. Unless there are units in the scenario that can attack bombers, this unit’s Defense factor is immaterial. (Example: Bomber)
- ▼ **Settler Unit** is the indispensable terrain improver and city builder. This unit normally has a zero Attack Factor. (Example: Settler)
- ▼ **Diplomatic Unit** allows all the complications that embassies, spying, and bribing can bring. This unit cannot attack, and it should have good movement and a moderate-to-low Defense Factor. (Example: Diplomat)

NOTE

This list includes only the bare functional minimum that every civilization should have available to them. Obviously, there are classes which are not covered here. Don't take this as implication that the others are any less interesting or useful. They're just less *essential*.

Keep in mind that, for all the characteristics cited in the list, the terms “low,” “high,” and so on are *relative to other units in the scenario*. If there are plenty of Armor units rolling around as the Attack Ground Unit, it would make no sense at all to have something like Warriors as the Defense Ground Unit.

Units and Advances

Like almost everything else about a scenario, there are no hard and fast rules about how to relate your units to the progress of research. If you're building a historical scenario with a limited time span, for instance, research might be ruled out altogether. At the other extreme, when you design a scenario that's a full-length game, the gradual appearance of better and better units is vital to keeping the player—your audience—interested.

As you might have noticed by now, I'm fond of summing up advice in a few bullet points. (For one thing, it saves you from trudging through lots of filler text to get to the point.) Here are my bullets for the interplay of units and advances:

- ▼ **Spread them out.** Don't let the units bunch up in any one area of the tech tree (unless it's intentional). Among other problems, that makes a few advances much more important than all the others, and players take advantage of choke points.
- ▼ **Units should progress.** Later units should be better than earlier ones—stronger, faster, with special abilities, or any combination thereof. The advance of science has a cost, and there should be a reward for that cost. It's possible to design a fun scenario in which this is not the case, but that's an exception, not the rule.
- ▼ **Old units should expire when new ones appear.** When you can build Riflemen, who wants the Production Menu cluttered up with Warriors and such? This doesn't mean that every new unit should make an existing one obsolete—maybe every third unit.

- ▼ **The tech tree is not a straight line.** Keep this in mind; players might not do research in the order you imagine. By focusing on one line of research, a player could get to certain units earlier than you expect. If the balance of your scenario depends on military science going a certain way, make sure there's no other way for players to go.
- ▼ **Obsolete units are irreplaceable.** If you want certain units to be unique, vitally important, or irreplaceable for any reason, make them obsolete at the start of the scenario. No one can build them, so the existing ones cannot be replaced. Exactly how you do this is important. Please refer to Chapter 10: Tricks of the Trade, for a detailed discussion.

A method I've found extremely helpful is to actually chart out the tech tree for the scenario, then add the units in a way that's easy to see from a distance. Once that's done, you can take a step back and examine the whole system at once. Looking at the “big picture” this way can give you a visceral feel for whether the progress of units in the scenario is balanced.

Charting a tech tree is no cakewalk. I cover my favorite method in Step 4: Build a Tech Tree. If you use that method, adding units to the chart is fairly simple.

TIP

Now, let's get on to the details.

Defining Characteristics

In this section, I review the format and content of a unit definition in some detail. The better you understand the unit entries in *rules.txt*, the more quickly you'll be able to implement the units you imagine—and the more interesting your scenarios can be.

If you have *Fantastic Worlds*, you can modify the units using the Unit Editor on the Editors menu. While this method is more convenient and probably easier for most folks, with a little experience, setting up your units manually is much faster.

**FANTASTIC
WORLDS**

Understanding the unit definitions is essential. The types of units in your scenario are defined in *rules.txt*. Search through that file until you find this line:

@UNITS

That's the header marking the beginning of the section containing the definitions of every unit type. Each of the lines of text following it—down to and including the line that begins with **Extra Air**—determines the characteristics of a single unit type.



If you have *Fantastic Worlds* installed, your *rules.txt* has 8 extra unit slots—Test Unit 1 through Test Unit 8.



All the lines that start with the @ symbol are header lines. Do not *ever* change *any* of the header lines. If you do, *Civilization II* will probably crash. Also, you should never add lines to or delete lines from *rules.txt*—not even empty lines. In some cases, the program counts lines to find data, and if you remove or add something, you will only cause trouble.

Just like we did for the terrain entries, let's take one of these definition lines apart. The Settlers unit is defined as:

`Settlers,Exp,0,1,0,0a,ld,2h,lf,4,0,5,nil,0000000000000000`

Each part of the line is a data field, and each field except the last one must end with a comma. Table 3-1 covers the meanings of all the fields.

Table 3-1. Unit Data Fields for Settlers

VALUE	FIELD NAME	EFFECT
Settlers,	Name	This is the name of the unit. It shows up in on-screen text. Like the manual says, names shouldn't be more than 15 characters, because they won't fit in all the display boxes and lists. Also, you should use only letters, numbers, and spaces in names; other characters might not be displayed and can occasionally cause program errors.

VALUE	FIELD NAME	EFFECT
Exp	Expires	After a civilization discovers this advance, it is no longer able to build this unit. Note that by giving this advance to every civilization at the start of the scenario, you can prevent the unit from being built at all. That way, the few you put in place are the only ones there can be—irreplaceable units. <i>Nil</i> in this field means that the unit never becomes obsolete.
0-2	Domain	Every unit exists and can move in one of three environments. The value in this field determines which is right for this type of unit. The three valid options are 0 (Ground), 1 (Air), and 2 (Sea). Any other number in this space causes problems. (If you have any trouble remembering the Domain numbers, try the mnemonic acronym GAS—Ground, Air, Sea.) For more, see the subsection Domains in this chapter.
1-5	Move	This is the movement allowance for ground and sea units, and it's the base from which the movement allowance for air units is calculated. Every mobile unit can move at least one square per turn, unless factors other than terrain interfere. If this is zero (0), the unit is immobile.
0-5	Range	For an air unit, this establishes how many turns it can stay aloft before it is destroyed. Note that the Move number for the unit is its per-turn movement allowance; the unit can move that many squares every turn it is aloft. If an air unit has a Range of zero, that unit can stay aloft indefinitely (like the Helicopter unit), but loses Hit Points every turn that it does not return to a city or air base (though it will never be destroyed by this loss). Note also that an air unit with a range of zero can capture cities. For ground and sea units, the Range is irrelevant and should always be zero.

VALUE	FIELD NAME	EFFECT
0a 1	Attack Factor	This is the base strength of the unit in battle when it is the aggressor, followed by the letter a . (Do not remove the letter.) In actual combat, many factors combine to determine the <i>effective</i> Attack Factor. If this is zero, the unit is unable to attack. An Attack Factor of 99 makes the unit a nuclear unit, regardless of any other characteristics. No civilization can build nuclear units until the Manhattan Project wonder (or the renamed equivalent) has been completed.
1d 1	Defense Factor	This is the base strength of the unit in battle when it is the defender, followed by the letter d . (Do not remove the letter.) In actual combat, many factors combine to determine the <i>effective</i> Defense Factor. Even if this is zero, the unit is not completely defenseless, due to a random factor in the combat calculations.
2h 1	Hit Points	This number is multiplied by ten to determine the amount of damage the unit can absorb before being destroyed. The number is followed immediately by the letter h . (Do not remove the letter.) This number should never be zero.
1f 1	Firepower	This number, which is followed by the letter f (do not remove the letter), determines how much damage the unit does to another unit when it scores a hit in combat. This number should never be zero.
4 1	Cost	This sets the cost in shields to build the unit. The number in this field is multiplied by the <i>Number of Rows in Shield Box</i> constant to get the actual cost of the building. (Thus, based on the default constant—10 rows—building the Settler requires 40 shields.) Make sure that this cost reflects how powerful and rare the unit is in your world. This field should never be set to zero.
0 1	Holds	For a sea unit, this determines how many other units it can carry at one time. Note that a sea unit can carry either ground units or air units, never both. (Which one a unit can carry is set by an attribute.) For ground and air units, this field is irrelevant and should always be zero.

VALUE	FIELD NAME	EFFECT
5–7	Role	This number determines how the computer-controlled civilizations (the AI) will use the unit. If this number is 5, 6, or 7, it also changes the unit's abilities. See the Roles subsection for more details. This field should never be anything but a number from zero (0) to seven (7).
nil	Prerequisite	A civilization must discover this advance—denoted here by the advance's abbreviation—before it can begin to build the unit. (This is a good reason why you should never change the official abbreviations of the advances.) <i>Nil</i> in this field means that the unit has no prerequisite, thereby allowing every civilization to build it from the beginning of the scenario (unless it is obsolete). <i>No</i> here prevents the unit from appearing in the scenario.
(zeroes),	Attributes	Various; these are discussed in detail in the Attributes subsection.

Domains

Every unit in *Civilization II* is assigned to one of three Domains. These are defined in the normal game as physical environments—ground, air, and sea—but in a scenario, a Domain is not necessarily a physical environment. As far as the program is concerned, each Domain is just a set of characteristics. An example will help demonstrate what I mean. A sea unit is defined as a unit which (among other things):

- ▼ Cannot move except on terrain squares of type 10 and friendly cities (but it can *exist* elsewhere).
- ▼ Can only be built in cities adjacent to terrain of type 10.
- ▼ Attacks units on terrain of types other than 10 at a Firepower of 1.
- ▼ Defends cities against units of other Domains at a Firepower of 1.
- ▼ Can be given the ability to carry units of one other Domain.
- ▼ Has its movement allowance increased by certain wonders.
- ▼ Cannot capture enemy cities.

The Table Method

For those of you who are experienced Microsoft Word users, there's a shortcut method you can use when redesigning units for your scenarios. If you convert the unit definition portion of *rules.txt* into a 14-column table, it makes it much easier to keep track of which field is which. Here's one way to do it (not necessarily the best):

- 1 Create a new Word document, and copy the text of the unit definition lines into the document.
- 2 Use the Replace feature to eliminate all the spaces.
- 3 Convert the text to a table, using the comma as the field separator. (You should end up with 14 columns.)
- 4 Put borders around the cells (if you want to—it helps me).

Now, do all your editing. When you're done and ready to put things back (or test your units), reverse the process:

- 1 Select the whole table and convert it back to text, separating the text with commas.
- 2 Select all the text and use the Copy feature to put that text into the Windows Clipboard.
- 3 Open *rules.txt* and find the @UNITS line.
- 4 Select all the old unit definition lines, then use the Paste feature to replace them with your edited text.
- 5 Save *rules.txt* as a Text Only file. (If it appears, don't believe the prompt about "formatting that cannot be saved in text format"—it's a crock. There's no non-text formatting necessary in *rules.txt*.)

There's a similar shortcut that could help experienced Microsoft Excel users, but it's more complicated. If neither of these is appropriate for you, don't worry. Editing the original text file is not really difficult; these shortcuts are just alternate ways of approaching the task.

Notice that there is no mention of sea, ocean, or water. The important fact about the different Domains is that the units in each have different characteristics, and the ways they interact with each other, with units of other Domains, and with cities are strictly defined—and all different.

Think about that for a while. You might be surprised at some of the ideas that spring up. Sometimes a different way of looking at a thing opens up whole new realms of possibility.

Roles

A unit can have one of eight different Roles, numbered from 0 to 8. Each Role is intended for use with units of a certain Domain. Units in other Domains can be given a Role not intended for them without crashing the game, but the result is always disappointing. It never makes the scenario more fun. Table 3-2 covers all the possible Roles.

Table 3-2. Unit Roles

ROLE	SHORT TITLE	DOMAIN	DESCRIPTION
0	Attack	Ground	The AI uses this unit primarily to explore territory, probe for and destroy enemy units, and to attack enemy cities. Attack units are also sometimes used as guards for Trade units, Settler units, and Diplomatic units.

ROLE	SHORT TITLE	DOMAIN	DESCRIPTION
1	Defense	Ground	The AI uses this unit primarily to defend its cities, both inside the city and in Fortresses nearby. Defense units might also be used to protect Trade units, Settler units, and Diplomatic units. When all cities are well guarded, Defense units might be used as attackers.
2	Naval Superiority	Sea	Naval units are nearly useless for city defense, so units with this role explore the oceans, destroy enemy shipping (especially transports), blockade coasts, and sometimes attack cities. These vessels also run in convoys to protect Sea Transport units and intercept air attacks.
3	Air Superiority	Air	This is the only Role for air units. The AI sometimes uses these units to patrol the airspace around cities. Much more often, however, air units fly out in front of a ground force to attack units of all Domains and damage or eliminate units in enemy cities.
4	Sea Transport	Sea	These units carry other units from place to place and are sometimes used for exploration. They seldom attack unless they have no units on board.
5	Settler	Ground	This gives ground units the basic settler abilities: establishing cities, adding to cities, and improving terrain—which is how the AI uses these units. Settler units <i>can</i> attack if their Attack Factor is not zero. A unit in slot 2 with this Role has the Engineer abilities.
6	Diplomat	Ground	This allows ground units to perform the basic diplomat functions: bribery, city investigation, sabotage, technology theft, and establishing embassies. The AI will use them as such. This Role also renders the unit unable to attack, regardless of its Attack Factor. (If you give this Role to an Air or Sea unit, it gains no diplomat skills, but does lose its ability to attack.) A unit in slot 47 with this Role has the Spy abilities.
7	Trade	Ground	This Role makes units suitable for establishing trade routes and helping to build Wonders of the World, and that's what the AI will use them for, along with exploration (especially inside enemy territory). Trade units <i>can</i> attack other units if their Attack Factor is not zero, but they cannot capture cities or fight units that are in cities. (If you give this Role to an air or sea unit, it has no effect.)

Why are they backwards?

Any of you who have already looked through the *rules.txt* file probably noticed that the explanation of attribute flags included in that file seems to be in reverse order. That's because those notes were written by a programmer. If you're counting in binary notation, which some programmers get in the habit of doing, those notes are in numerical order. In binary, 000000000000001 equals 1, 000000000000010 equals 2, 0000000000000100 equals 4, and so on.

That's not exactly how the program sees it, of course. In programming lingo, the attribute listing is a set of 15 "Boolean bits". Each bit—the digit in that position—is in one of two states: *True* = 1 or *False* = 0. (You could also call the states *On* and *Off*.) Taken together, these 15 bits (and one other bit the function of which would take me a while to explain) make up what programmers call a single 16-bit "word". It's a fairly economical way to represent information, and good programmers like to be thrifty with memory.

Since we're human beings, though, and not programs, there's a better way to think of the attribute flags. Each one is simply a switch that is either on (1) or off (0). The combination of all the switches determines all the attributes for the unit.

For those of you who are used to reading from left to right (instead of in binary), Table 3-3 lists the attributes in the *other* direction.

Attributes

The last piece of data in each of the unit definitions is a long line of zeros and ones. These fifteen digits are the "attribute flags" that tell *Civilization II* what special abilities—if any—each type of unit should have. Using these is a great way to spice up a scenario, but you've got to watch your step.

To edit one of these attributes, you simply change the zero to a one or vice versa. Be careful that you get the digit you want—mistakes here can cause weirdness and game crashes. (This is where the table method really comes in handy.)

FANTASTIC WORLDS

If you're using the Units Editor, once you've selected the unit you want to mess with, click on the Abilities button to change the attribute flags. You won't see any ones and zeros, but the entries in the on-screen checklist match the list of possible attributes.

Table 3-3. Unit Attributes from Left to Right

POSITION	EXAMPLE	SHORT DESCRIPTION
1	100000000000000	Can spot submarines
2	010000000000000	Double defense versus air units
3	001000000000000	Destroyed after attacking
4	000100000000000	Costs no shield support under Fundamentalism
5	000010000000000	Double defense versus mounted units
6	000001000000000	Treats all terrain as road

7	000000100000000	Can be ordered to paradrop
8	000000010000000	Can refuel air units
9	000000001000000	Ignores city walls
10	000000000100000	Might be lost away from land
11	000000000010000	Can attack bombers
12	000000000001000	Has the submarine characteristics
13	000000000000100	Can make amphibious assaults
14	000000000000010	Ignores zones of control
15	000000000000001	Can see for two spaces

Now let's expand on the short descriptions in Table 3-3 and in the *rules.txt* file. (Sometimes, the details can make a difference.)

Can spot submarines



This seems pretty straightforward (a lot of these do, but in some cases it's misleading). A unit with this attribute turned on ignores the stealth advantage of submarines and can spot them just like it can any other unit. This ability is not limited to spotting the original Submarine unit or the unit defined in the same unit slot as the old Submarine. In this case, "submarine" means any unit with attribute 12—Has the Submarine Characteristics—turned on.

This attribute might seem like its utility is limited to naval units and some air units, but that's not the case. Land units along the coast can't see subs, either, unless they have this ability. If you want a coastal defense unit that acts as a Ship Spotter, you should consider allowing it to spot subs, too.

Double defense versus air units



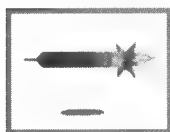
The flag in position 2 is exactly what it sounds like. When attacked by an air unit (any with a domain of 1), a unit with this attribute has its Defense Factor doubled before the combat calculations begin. Note that this has no effect when the unit attacks an air unit; it's strictly defensive.

What sort of units is this appropriate for? The answer depends on the air units in your scenario. For example, in the original *Civilization II*, only the AEGIS Cruiser had it—partly because this attribute gives extra protection from missiles, too. In the

Age of Reptiles Scenario, however, there's nothing like a missile, so I felt comfortable giving it to any dinosaur with enough armor (especially on its back).

You should not give this attribute to an air unit—it's pointless. After all, nothing but other fliers will ever attack an air unit. Doubling the unit's Defense Factor has exactly the same effect, and if you need to, you can adjust that later—to triple, one and a half, or whatever fits your scenario best.

Destroyed after attacking



As with many of these, the name pretty much says it all. Any unit with this attribute turned on is a one-shot weapon—it can attack only once and is immediately destroyed when the combat is over. Note, however, that the unit can *be attacked* multiple times. Defending does not count as an attack.

There's one undocumented effect to watch out for with this attribute. If you give this to an air unit (any unit with a Domain of 1), that unit is considered a *missile*. That means that units with the submarine characteristics (attribute 12) can carry this unit.

The use of this attribute is clear in the normal Cruise Missile and Nuclear Missile units. When building your scenarios, keep in mind that there are countless one-use, disposable weapons—snowballs, bee stings, kamikaze fighters, magical fireballs, and golems, just to name a few. What is important in the game, though, is that a one-shot be worth building. Missiles are devastating attackers, and therefore a player will build them even despite the cost and lack of reuse. If your one-shot units aren't strong, they should be cheap, or else have some other characteristic that makes them *very* attractive to build.

Costs no shield support for Fundamentalism



Whatever you call the type of government that functions like Fundamentalism (you can rename them—see Step 6: The Foundations of Civilization), the cities of any civilization that is ruled by that type of government do not pay the normal shield cost for supporting units with this attribute. Note, however, that the cities still pay any food support cost (as for Settlers).

This can be a convenient way to get around the problem that arises when you want to mount a horde of units from a small city, but there are consequences to be wary of. If it is possible in your scenario to produce more of these units, human players

will certainly take advantage of the fact. (Frankly, Fundamentalism itself is a problem in this respect, but I discuss that elsewhere.) If you choose to use this attribute at all, use it sparingly and wisely. Support-free units are a threat to the balance of any game.

A better way to free cities from the support burden of the units you place at the beginning of your scenario is to edit each unit individually and change its Home City to *None*. That can take some time, but it's worth it to avoid other problems.

Double defense versus mounted units



This attribute has exactly the same effect as attribute 2, *Double defense versus air units*, except that it applies to certain ground units (rather than air units). When attacked by a mounted unit, a unit with this attribute has its Defense Factor doubled before the combat calculations begin. Note that this has no effect when the unit attacks a mounted unit; it's strictly defensive.

How do you know which are the mounted units? Good question. A mounted unit is defined as any ground unit that is capable of attacking and has a movement allowance of 2. So you don't have to look them up, here's the list:

Horsemen

Chariots

Elephants

Crusaders

Knights

Dragoons

Cavalry

Howitzer

These all have some animal as their mode of propulsion (howitzers have at times been pulled by horses), so it's pretty clear why they're considered mounted units.

Engineers, Diplomats, and Freight all have a movement allowance of 2, but they can't attack, so they're not considered mounted units.

NOTE

Pikemen have this attribute because of the great defensive advantage they enjoy against mounted troops—when their pikes are braced against the ground, they present oncoming cavalry with a wall of spearpoints. What this is to you, however, is a chance to set a group of units apart in some way—by giving them all a movement allowance of 2—and to give certain units a defensive advantage against only those particular units. I'll just mention one example and leave the rest to your imagination.

Say you're working on a wild west scenario and you design several units armed with pistols—gunslingers and lawmen. You make those units “mounted” by setting their Move at 2. At some point in the game, civilizations can research rifles and, later, bulletproof vests. Both of these would confer an advantage on the new units they made possible, but the vest would give a purely defensive bonus, while the rifle has an offensive component, too. Vest-equipped units, then, could have the “*Double defense versus pistol units*” attribute.

As the original game is set up, there's no reason to give this attribute to air or naval units (none of the mounted units can attack fliers or ships). I've not yet found a way to test it (try it; it's a difficult situation to set up), but I believe that this attribute is not functional for non-ground units.

Treats all terrain as road



Attribute 6 can be defined two ways, and both *seem* to mean the same thing. You can say that for movement purposes, units with this attribute treat every terrain square as if it had a road on it, or you can say that these units use only one third of a movement point to move into any terrain square. The difference comes in when you change the Road Movement Multiplier (see Step 6: The Foundations of Civilization for the details).

The truth is, this attribute is what it says—the unit treats all terrain as if it had a road on it. (Railroads are the exception to this rule; like any other unit, units with this attribute use no movement points at all to move on railroads.) If you change the multiplier, you add to this unit's ability to move. With the default multiplier, the end result is to triple the movement allowance for the unit. (When you consider rough terrain, this really more than triples the unit's movement.) If you change the multiplier, though, the unit could become able to move *any number* of squares per turn.

This is not to be used lightly. While beginning players often call for faster units, that's only inexperience talking. Never mind the number of square miles that a terrain square represents, and it's immaterial that each turn could represent months or years. The important point is *game balance*. Fast units are appropriate if and only if

it makes the scenario more fun or more challenging—that's it. Nothing else matters. After all, the original game is addictive and fun despite the fact that it takes an ultra-modern Stealth Fighter several game years to go around the world.

If you give this movement advantage to a unit, you should balance that with a disadvantage in some other area—make it expensive to build or support, for example, or weak in attack or defense.

Perhaps it goes without saying, but this attribute is totally useless for air units. In fact, it could cause problems with the program. No matter how you change things, you should never give this attribute to an air unit.

Can be ordered to paradrop



This attribute marks the unit as one that is able to receive Paradrop orders. That's simple enough; a paradropping unit that begins its turn in a city or air-base (or your scenario equivalents thereof) can instantly move to any square in range that isn't occupied by an enemy unit. When it gets there, the unit has one movement point left.

Changing the range for paradropping is discussed in Step: 6 The Foundations of Civilization, as is the prospect of renaming both air bases and paradropping itself.

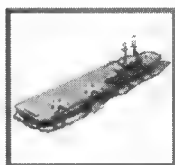
NOTE

This attribute is a lot of fun to play with, but it can be dangerous. One of the common conquest tactics used by *Civ II* players is to drop a Nuclear Missile on a city, then during the same turn (while the city is still empty of defending units) drop Paratroopers in to take over. If you use this, find a way to limit the power of the units. For example, you can make sure that nuclear weapons are expensive, make nuclear defenses cheap, limit the range of paradrops, keep the attack and defense of the paradropping units low, or design the game so that all civilizations get these units at roughly the same time. (*Civilization II* does all of these.) In the Mars Now! scenario, the LMO Troopers unit (LMO stands for Low Mars Orbit, by the way) can drop nearly anywhere on the planet—the paradrop range is 75 squares! However, the two nuclear units are difficult to get and come quite late in the scenario, which effectively nullifies that particular tactic.

In your scenarios, paradropping doesn't need to be defined as a physical movement. You can call it by any name—teleporting, burrowing underground, magic, or any other instant mode of travel that doesn't cross the intervening distance the normal way.

In the normal game, there's no reason to give this attribute to air units or naval units, but that might not remain true once you start changing things around. Still, you should only use this one with ground units, and not with any unit that has a non-zero Domain. We wouldn't want your scenario to crash, would we?

Can refuel air units



Attribute 8 is not exactly what the description in *rules.txt* makes it sound like. If you give this attribute to a naval unit (one with a Domain of 2), *and* give that unit at least one Hold in which to transport units, then you create an aircraft carrier—a unit that “can carry air units” (and *cannot* transport ground units).

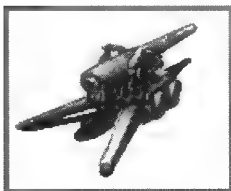
However, if the unit has no Holds, it can't carry units at all. This attribute would more correctly be named *Can refuel air units*.

What this really does is create a mobile refueling station. Note that you *can* give this attribute to a non-naval unit (ground or air). Here's the exact scoop:

- ▼ Any air unit that moves onto the same square with this unit (the “carrier” unit) ends its movement for that turn at that square.
- ▼ The air unit has its range reset—is refueled, if you prefer—as if it had landed at a city, an airbase, or a Carrier.
- ▼ If the carrier unit moves before the air unit leaves the square, the air unit remains behind; the carrier unit does not transport air units around unless it has at least one Hold (and is a naval unit, since only naval units can carry other units).

So, to answer all of those *Civ II* fans who have written in demanding that we put midair refueling in the game—you already have it, and you've had it all along.

Ignores city walls



This one is simple; if a unit with this attribute attacks a city that has city walls, the units defending that city do not get the tripling of their defense that they normally would. This applies only to attacks by this unit—it has no lasting effect on the city or the units defending it.

The Howitzer unit has this attribute to simulate the fact that real-life howitzers can fire their projectiles over city walls. In the Age of Reptiles scenario, most of the archosaurs and saurosaurs, because of their great size, ignore the effect of Ramparts. Depending on what you name the City Walls improvement in your scenario, the rationale for giving this ability to a unit could be nearly anything.

While building scenarios, I've noticed that the Great Wall wonder—no matter what it's called in the scenario—can easily upset the balance of the game if it's available too early, lasts too long, or is too inexpensive to build. One way to help restore the balance is to make units with this attribute available earlier than in the normal game.

TIP

This attribute is totally useless for air and naval units, because the effects of City Walls are not applied to their attacks in the first place.

Might be lost away from land

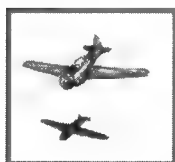


We all know how Triremes work; whenever they end a turn on a square that's not adjacent to a land square, there's a fifty-fifty chance that they'll sink. (The chances of this happening are actually under your control; see Step 6: The Foundations of Civilization.) This attribute is what makes that happen. Any naval unit (this attribute has no effect on ground and air units) with this attribute turned on is not cut out for travel on the open sea.

It's not necessary to restrict this to less-than-sturdy types of ships. The immense sauropods in the Age of Reptiles scenario and the powerful ironclads in the American Civil War scenario were both limited to near-shore operations—one because they couldn't swim and the other due to buoyancy and oxygen supply issues.

Among other uses, this is a great way to limit the travel options of civilizations in your scenarios. If you space the land masses correctly, you can force seafarers to use only certain routes—or prohibit contact entirely. Most players will rarely risk units in attempts to cross at “forbidden” points, especially if the chance of losing the unit is high.

Can attack bombers



In the default game of *Civilization II*, only Fighters and Stealth Fighters can attack Bombers and Stealth Bombers. All air units have air-to-air capabilities, but because most air units have a Range of 1, there's never a chance to attack them—enemy planes other than bombers are never in the air during your turn. The fact is, any air unit can attack any other air unit, unless the unit being attacked

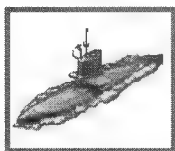
has a Range greater than 1—and thus is a “bomber”. You can make an air unit able to attack “bombers” using attribute 11. Any air unit (Domain of 1) with this attribute turned on can attack other air units with Ranges above 1.

As I explained earlier, you don’t have to see the three domains as physical; they can be viewed instead as sets of characteristics. One of the characteristics of the units you’re putting in the “air” domain is that they cannot be attacked except by units in the same domain, and those that also have a Range >1 are only vulnerable to those which also have this attribute. (Of course, air units can also be damaged and defeated by any unit they attack.)

Let’s settle for just one example. You could decide that units in domain 1 are “ethereal” and only special ethereal units—spirit predators or ghostly hunters of some sort—can attack those that are able to travel far from their physical bodies. Substitute the word “subterranean” or “cloaked” or “interdimensional” for “ethereal,” and you start to see the possibilities.

This attribute has no effect if you give it to a naval or ground unit, but it won’t crash the game, either.

Has the submarine characteristics



This attribute is only applicable to naval units (those with a Domain of 1). If applied to any other type of unit, it has no effect.

If you’ve looked in *rules.txt*, you know that this attribute flag is noted there as giving “Submarine advantages/disadvantages” to the unit. That’s rather vague, but only because this attribute flag has too many effects to describe any other way. When this flag is on, the unit has all of the special abilities and weaknesses assigned to the original *Civ II* Submarine unit. Here they are:

- ▼ The unit is invisible to all units which do not have attribute 1, *Can spot submarines*, turned on. It remains unseen until it attacks or is attacked by accident (bumped into).
- ▼ No matter what Defense Factor is assigned to the unit, when attacked it always defends with a factor of 1, even if it’s attacked by another “submarine” unit.
- ▼ The unit cannot attack ground units (those with a Domain of 0), and therefore is of no direct use against cities.
- ▼ The unit can carry (and launch) missiles.

For the purposes of this attribute, a “missile” is defined as any air unit that has attribute 3, *Destroyed after attacking*, turned on.

NOTE

When you think about it, there’s no good reason this attribute should be limited to naval units. (Think of a land-based, camouflaged, mobile missile launcher with little defensive capability.) However, those of you who have already read Step 2: Create Your Map, know that if land-based submarines are important enough to you, you can reverse the land and sea domains.

Can make amphibious assaults



This Attribute is limited in the original game to the Marines unit. This is meant to emphasize the difficulty involved in landing and securing a beachhead; it is not really a game balance issue. You can assign this attribute to any ground unit, and it will be able to attack another ground unit (or a city) from on board a naval transport vessel.

This does become a game balance issue in scenarios that limit seagoing power to certain civilizations. If the empire under attack does not have naval units, there’s no hope of intercepting the amphibious assault before it reaches the coast. The assaulting units need not land outside a city unless there are no cities located on the coast; thus the attack cannot normally be intercepted by ground units, either. (The AI wouldn’t know that it was at such a disadvantage, and so would not avoid building cities along the coast.) This makes in-city defenses extremely important. Unless a situation of this type is your intention, you should avoid creating the combination of ample amphibious units and overwhelming naval power.

One idea that I can’t help mentioning is the combination of this attribute with attribute 3, *Destroyed after attacking*. This would seem to make for a perfect Cannonball unit or other sea-to-shore projectile. Unfortunately, it would be possible for the Cannonball to go gallivanting around the countryside like any other ground unit. Unless all the land masses in your scenario are tiny, one-square islands, it won’t work.

This attribute might be perfect for marauding Vikings, but it has no effect when given to naval units (since they cannot carry one another). Air units can attack directly from the carrier already, so they have no need of this ability.

Ignores zones of control



Here's one attribute that has already caused some trouble. In the original release of *Civilization II*, the Engineer unit had this attribute turned on, and thus had the ability to move past enemy units and cities unhampered by their zones of control. This made it far too easy for an enemy empire to sneak an Engineer deep into someone's territory and establish a city right next door to the capital. One of the first patches MicroProse released fixed this problem.

That's an example of the trouble you can get into; this attribute can seriously unbalance a scenario by allowing a dangerous latitude of movement to a specific type of unit. That's why this ability is normally reserved for units that are weak and have a specific function to perform that would be impossible (or nearly so) without it—Spies, Diplomats, Explorers, and the trade units, Freight and Caravan.

You might also find this one appropriate for units that are difficult to stop—extremely fast units, for instance—but if you design the unit's combat characteristics well, you'll normally find that this ability becomes unnecessary.

This attribute (like attribute 9, *Ignores city walls*) has no effect when applied to air and naval units. That's because zones of control in *Civilization II* are not enforced in the air or on the ocean.

Can see two spaces



This is one attribute that is exactly as simple as it seems. When this one is turned on, the unit can see twice as far as normal units see.

NOTE

If you read the note in *rules.txt*, it refers to "two-space visibility." What it means by "visibility" is how far the unit can see—as in a pilot saying, "Visibility is fifteen miles today"—not how far it can be seen.

Air units, for instance, have a greater view due to their altitude. Very tall units (the saurosaurs in the Age of Reptiles, for example) might also have a better range of vision. Generally, there's not much risk that this attribute will unbalance your scenario, unless there are things you wish to hide or you only give it to one civilization's units. A good general rule is to give this ability to a unit only if it makes sense for that unit to be able to see farther than most.

Changing Icons

When you've gone to all the trouble of defining exciting new units, naturally you want to make the look of each unit match its function. In every scenario—whether it's historical or fantastic, an accurate uniform or an alien life-form—part of the fun for the player is getting a first look at the cool new units. It's not difficult to make new icons, but you need a way to edit GIF files.

If you have *Fantastic Worlds*, you can modify the looks of every unit icon using the Icon Editor in the Units Editor on the Editors menu. This method is more convenient and easier than a GIF editor for most folks.



NOTE

For more info on the GIF format, read the section on GIF Editors in Step 1: Getting Started.

The remainder of this section assumes that you have a GIF editor or *Fantastic Worlds*.

All the unit icons (plus a few other things that can be useful) are contained in the file *units.gif*. (See Figure 3-1.) Once you understand the way this file is organized and how to find the icon you're after, you should have no trouble editing the looks of units.

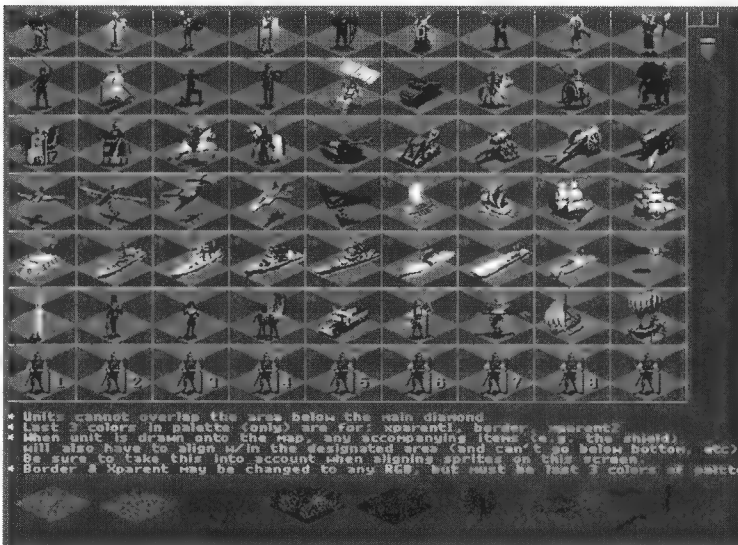


Figure 3-1: *Units.gif* contains all the unit icons.

The Palette and Green Lines

Files in GIF format are one example of what are called *paletted* graphics files. The palette (sometimes called a “color table”) is a collection of colors defined for use in the file, and no other colors appear in that file. Most graphics programs include a way to display the palette, usually as a box full of colors you can pick from. All of the files for *Civilization II* use the same palette, and you should never change the palette in any of the files—ugly graphic problems result, and the game could crash, too.

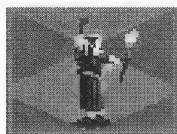
The *Civ II* palette contains some colors that are recognized by the program as “not to be displayed” colors. These are mostly “transparent” colors that we can use in the files to help us position graphic elements correctly. For example, the unit icons all sit on a purplish background in the shape of a normal terrain square (also called a “tile”). That purplish diamond sits on a gray background. Both the purplish color and that exact shade of gray are transparent colors. When the unit icon is displayed on terrain, you can still see the terrain underneath—in every place that the purplish color and the gray appear in the graphic file. The purple tile gives us a way to know where on the terrain tile the icon will appear.



CAUTION

Some colors in the palette are very similar to the transparent colors, especially the transparent shade of gray. If you get them mixed up, your game will look pretty bad, but probably won't crash. Just re-edit the file and replace the wrong color with the right one, and the problem goes away.

For example, here's how Fanatics look in the file:



and how they look in the game:



Unit icons can overlap the upper gray areas, but should *never* cross over the lower borders of the purple diamond.

Another color that you must be careful about is the bright green used to draw all the boxes around the icons. Those green boxes are used by the program as reference to

tell where the icons are. Every unit icon is in a box of exactly the right size and shape, and you should never mess with the green lines. Work inside the lines.

Never move or erase any of the green lines. Don't add new ones, either.

CAUTION



Active Areas

If you look in the *units.gif* file, you'll notice a few graphics and other markings that obviously aren't units. The MicroProse artists left some tools in the files, plus there are some bits that the program uses. Anything that does not appear in the game is in an *inactive area* of the GIF file. Anything outside of the *active areas* (see Figure 3-2) is either ignored by the game program or is something you should never touch, so there's no point in changing any of it.

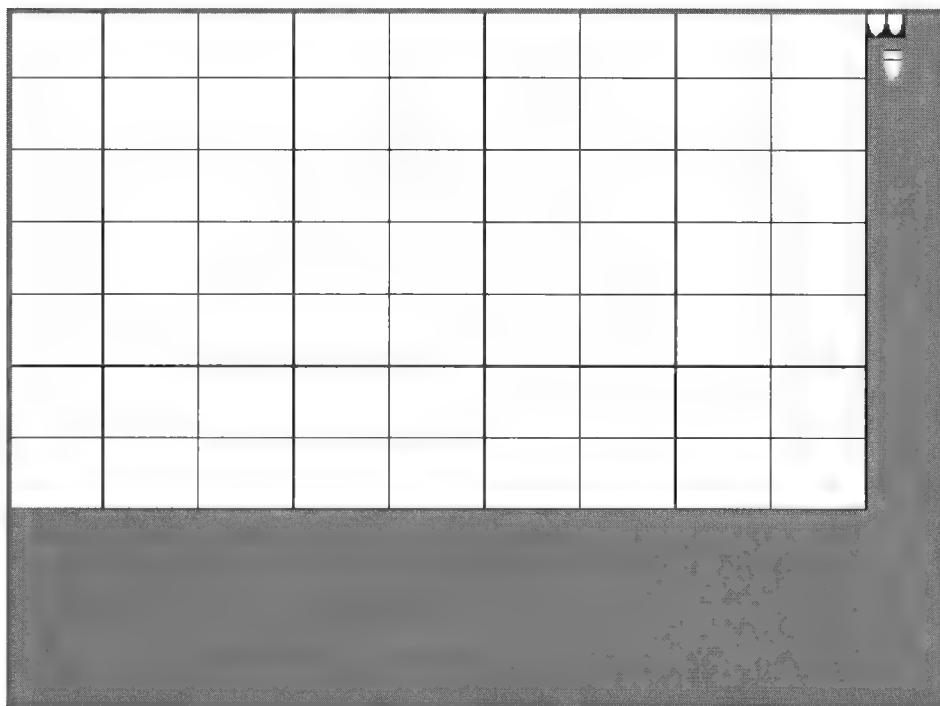


Figure 3-2: Active areas of *units.gif*

For example, along the bottom of the file is a line of terrain squares. Those aren't used in the game, so they're inactive. (What are they there for? Read on.)

Editing an Icon

Changing a unit icon is a simple matter of modifying the picture that matches the unit slot. Every icon corresponds to one of the unit slots in *rules.txt*. They're in order, but if you're like me and would rather not count, Figure 3-3 might come in handy.

Figure 3-3: Each icon corresponds to a unit slot.

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53
Barb. Ldr.	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
54	55	56	57	58	59	60	61	Barb. Ldr.

Thus, for example, if you copied the entire Armor icon (purple and gray areas included) and pasted it over the Fighter icon, in the game you'd have a flying tank that could attack all types of units, but needed to return to a city or air base to refuel every turn. It's the placement, not the icon itself, that matters.

There are a few guidelines to keep in mind when you're making unit icons, but for the most part, it's a free-for-all. Do what you want, but:

- ▼ Don't use the transparent colors (the green, purple, and gray), except in places where you *want* the terrain underneath to show through.
- ▼ Don't use colors that you also use in your terrain icons, unless you *want* the icon to become partially invisible on a specific type of terrain. This is when the terrain squares at the bottom of the file come in handy as a reference.

- ▼ Don't go outside of the green-bordered icon box.
- ▼ Do not ever allow an icon to cross the lower borders of the purple diamond. Put nothing in the lower gray areas.

You might also consider leaving room in the box for the unit's shield.

The Shield Pixels

There is one (and only one) circumstance in which you should mess with the green lines that define the unit boxes. That's when you place the *shield pixels*.

If you look closely, you'll see that there are two blue dots associated with each unit icon. One is somewhere in the green line that defines the top of the unit box, and the other is in the green line along the left side. These two, taken together, tell *Civilization II* where to place the shield that every unit carries.

The pixel on the top defines the leftmost edge of the shield. (For this reason, it's sometimes called the "left pixel," but that gets confusing.) Make sure that this point is far enough from the right side of the unit box, or the shield will overlap the box and cause tracers—the overlapping portion of the shield is not erased—when the unit moves. For the same reason, never put this pixel in the leftmost two pixels of the box, either.

The pixel on the left defines the top edge of the shield. (Therefore, it's sometimes called the "top pixel," so you can confuse it with the other one.) Make sure that this point is far enough from the bottom of the unit box, or the shield will overlap the box and cause tracers—the overlapping portion of the shield is not erased—when the unit moves. For the same reason, never put this pixel in the upper two pixels of the box, either.

Whenever you change the look of a unit, you should determine where the shield looks best, then use these pixels to put it there.

The Icon Editor in the Unit Editor has a tool for placing the shield that takes care of the pixels for you.

FANTASTIC
WORLDS

If there are no shield pixels associated with a unit, the program places the shield in the upper left-hand corner of the box, and when you play the game, that unit will leave tracers (the unerased left and top edges of the shield) when it moves.

Do Not Touch

Here's a general rule that I apply to all the *Civ II* files: If you don't know what it is and how it works, don't touch it.

When it comes to the units file, there are only a few parts that neither you nor I nor anyone else should change: the ones I've already mentioned—the green lines, the purple tiles, and the lower gray areas. You can change the shape of the shield outline, but I don't recommend it (and if you do, remember to also change the shields in *icons.gif* to match *and* don't mess with the size and shape of the Hit Points bar).

NOTE

When you're editing *units.gif*, you'll see some purple (and therefore transparent) letters in the upper gray portions of some of the unit icon areas. Those are the initials of the MicroProse artist who created the icon.

Understanding Unit Sounds

Giving units new sounds is easy and fun, but unless you have *Fantastic Worlds*, it's difficult to figure out which units use which sound files. This subsection covers how sounds are assigned to units, how to determine exactly what units are using what sounds, and how to replace the existing sounds.

Who Says What?

First of all, let's establish what sound or sounds *Civilization II* plays for each unit. If you've ever tried to figure it out yourself, you know how difficult it can be to understand the pattern of unit sounds. It's not exactly user-friendly.

- ▼ Most of the sounds are connected with combat. Any unit that can attack makes a noise when it does so. Some make two or even three different sounds (like the Howitzer). Note also that attack sounds might play multiple times—especially when naval units battle.
- ▼ With a few exceptions, the attack sounds are based on the unit's Domain first, then on some other criterion or criteria.
- ▼ Often, the slot that the unit is in (along with the Domain) determines what sounds it uses.

- ▼ Certain other characteristics, like Range and Role, can change the unit's sound.
- ▼ Certain attributes, too, can affect the unit's sound choice. For example, any unit with the *Destroyed After Attacking* attribute uses the missile sound and no other.
- ▼ Air units have extra sounds that others do not: crashing, running out of range, and two types of attack—versus another air unit and versus any other type.
- ▼ If a unit is nuclear, there is only one sound for it, no matter what its Domain, slot, characteristics, or attributes.
- ▼ There are also a few sounds that are special and very specific—the sound the Freight unit makes when it reaches its destination, for example.

Rather than try to explain every possible case and exception, I've compiled them into Table 3-4. As you read over the table, you'll notice that some of the sounds are used by quite a variety of units. These are the "general-use" sound files. It's wise not to put specific sounds in these files unless you've strictly limited the units the sounds are applied to. After all, the sound a unit makes should be appropriate to the unit. An ultramodern battle tank shouldn't sound the same as a motley band of barbarian warriors, but if you're not careful, they will.

Keep in mind that the general-use sounds need only be appropriate for the units that exist or can be built in your scenario. You do not need to consider any unit that you have eliminated.

NOTE

Table 3-4 lists all of the sound files any unit in *Civilization II* might make and the circumstances that cause that sound to be played. The explanation includes exactly what types of unit use the sound.

Table 3-4. Unit Sounds

SOUND FILE	PLAYS WHEN...
<i>Aircombt.wav</i>	A non-missile air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot attacks another air unit.
<i>Biggun.wav</i>	When any naval unit attacks, other than those in slots 37 through 40 and any with special sounds (<i>torpedo.wav</i> and such).
<i>Boatsink.wav</i>	A naval unit is lost at sea.
<i>Catapult.wav</i>	A ground unit in slot 23 (the Catapult position) attacks.

<i>Cavalry.wav</i>	A ground unit in slot 20 (Dragoons) or slot 21 (Cavalry) attacks.
<i>Custom1.wav</i>	The unit in slot 51 (Extra Land) attacks.
<i>Custom2.wav</i>	The unit in slot 52 (Extra Ship) attacks.
<i>Custom2.wav</i>	The unit in slot 53 (Extra Air) attacks.
<i>Diesel.wav</i>	A trade unit (Role = 7) in slot 49 (Freight) arrives at its destination and fulfills one of its functions.
<i>Divcrash.wav</i>	An air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot is destroyed in combat.
<i>Divebomb.wav</i>	A non-missile air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot attacks a city, a ground unit, or a naval unit.
<i>Elephant.wav</i>	A ground unit in slot 17 (Elephant) attacks.
<i>Engnsput.wav</i>	An air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot is destroyed for exceeding its range.
<i>Extra1.wav</i>	The unit in slot 54 (Test Unit 1) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra2.wav</i>	The unit in slot 55 (Test Unit 2) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra3.wav</i>	The unit in slot 56 (Test Unit 3) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra4.wav</i>	The unit in slot 57 (Test Unit 4) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra5.wav</i>	The unit in slot 58 (Test Unit 5) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra6.wav</i>	The unit in slot 59 (Test Unit 6) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra7.wav</i>	The unit in slot 60 (Test Unit 7) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra8.wav</i>	The unit in slot 61 (Test Unit 8) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Feedbk04.wav</i>	A diplomatic unit (Role = 6) is expelled from enemy territory.
<i>Fire---.wav</i>	A ground unit in slot 24 (Cannon), 25 (Artillery), or 26 (Howitzer) attacks. This sound is normally followed by <i>Medgun.wav</i> , then <i>Largexpl.wav</i> .
<i>Helishot.wav</i>	An air unit with a Range of zero (0) attacks.
<i>Infantry.wav</i>	A ground unit in slot 7 (Musketeers), 9 (Partisans), 10 (Alpine Troops), or 11 (Riflemen) attacks.

<i>Jetbomb.wav</i>	A non-missile air unit in slot 30 (Stealth Fighter) or any higher-numbered slot attacks a city, a ground unit, or a naval unit.
<i>Jetcombt.wav</i>	A non-missile air unit in slot 30 (Stealth Fighter) or any higher-numbered slot attacks another air unit.
<i>Jetcrash.wav</i>	An air unit in slot 30 (Stealth Fighter) or any higher-numbered slot is destroyed in combat.
<i>Jetsputr.wav</i>	An air unit in slot 30 (Stealth Fighter) or any higher-numbered slot is destroyed for exceeding its range.
<i>Largexpl.wav</i>	As the final sound when any naval unit attacks. This is also the final sound when a ground unit that plays <i>Medgun.wav</i> (slots 22, 24, 25, and 26) attacks.
<i>Mchnguns.wav</i>	A ground unit in slot 8 (Fanatics), 12 (Marines), 13 (Paratroopers), or 14 (Mech. Inf.) attacks.
<i>Medexpl.wav</i>	Despite what its name implies, this sound is not used.
<i>Medgun.wav</i>	A ground unit in slot 22 (Armor) attacks. This sound is also played after <i>Fire---.wav</i> (for ground units in slots 23 through 25).
<i>Missile.wav</i>	Any unit with the <i>Destroyed after attacking</i> attribute attacks. This overrides <i>any</i> other sounds the unit might be eligible for except <i>Nukexplo.wav</i> .
<i>Movpiece.wav</i>	The player moves a unit.
<i>Navbttle.wav</i>	A naval unit in slot 37 (Destroyer), 38 (Cruiser), 39 (AEGIS Cruiser), or 40 (Battleship) attacks. This sound is normally followed by <i>Largexpl.wav</i> .
<i>Neg1.wav</i>	The player tries to move a unit where it cannot go.
<i>Nukexplo.wav</i>	Any unit with an attack factor of 99a attacks. This overrides <i>any</i> other sounds the unit might be eligible for.
<i>Smallexp.wav</i>	Despite what its name implies, this sound is not used.
<i>Spysound.wav</i>	A diplomatic unit (Role = 6) successfully bribes a unit, investigates a city, establishes an embassy, or incites a revolt.
<i>Swordfgt.wav</i>	A ground unit that does not fit the criteria for any of the other ground unit sounds attacks. (For the exceptions, see <i>Catapult.wav</i> , <i>Cavalry.wav</i> , <i>Elephant.wav</i> , <i>Fire---.wav</i> , <i>Infantry.wav</i> , <i>Mchnguns.wav</i> , <i>Missile.wav</i> , <i>Nukexplo.wav</i> , and <i>Swrthors.wav</i> .)
<i>Swrthors.wav</i>	A ground unit in slot 15 (Horsemen), 16 (Chariot), 18 (Crusaders), or 19 (Knights) attacks.
<i>Torpedos.wav</i>	A naval unit with the Has Submarine Characteristics attribute attacks.

Replacing Sounds

To actually change the unit sounds in your scenario, you simply insert a sound file of your own—with the exact same name as the file you want to replace—into the Sound subfolder. When the time comes to play the sound, the program searches there for the file with the appropriate name, then plays it.



With *Fantastic Worlds*, you can use the Sound Editor sub-function of the Units Editor to change your units' sounds. The really useful thing about this is that the editor shows you exactly which sounds the unit is using, based on the current characteristics and attributes you've set. The first problem is that when you replace a sound for one unit, the editor doesn't notify you that you're changing it for every unit that uses the same sound file. The second is that the editor doesn't cover *all* of the unit sounds. That's why you need Table 3-4.

The only catch is that *Civ II* sounds need to be in a specific format. (I'll go over these specifications again in other chapters, in case some of you are skipping around the book.) For *Civilization II* to be able to play a WAV file, it should be in this format:

- ▼ **22kHz**—This is the only sampling rate that's guaranteed to work. If you try to use anything else, you're on your own.
- ▼ **8-bit**—The game works well with 16-bit sounds, but that's *only if* you're not using a pre-95 version of Windows *or* you have a 16-bit sound card. Versions of Windows earlier than 95 (3.1x) cannot play 16-bit sounds on an 8-bit sound card.
- ▼ **Mono**—Once again, if you have Windows 95 (or later) or a 16-bit sound card, you can probably use stereo sound effects without a problem. Otherwise, forget it.

Why all the strictures and the low quality sound? Remember, *Civilization II* was developed to work with Windows 3.1 (and anything newer, of course). Sound capabilities have advanced quite a bit in the last couple of years. ☒

STEP 4

BUILD A TECH TREE

IT'S NOT TOO FAR OFF THE MARK to say that research is the core of *Civilization II*. Other game factors are important, but science is vital. In scenarios, that is not necessarily true. The X-COM: Assault scenario, for example, includes no research at all. The majority of scenarios, however, depend on science to be the essential factor that it is in the original game.

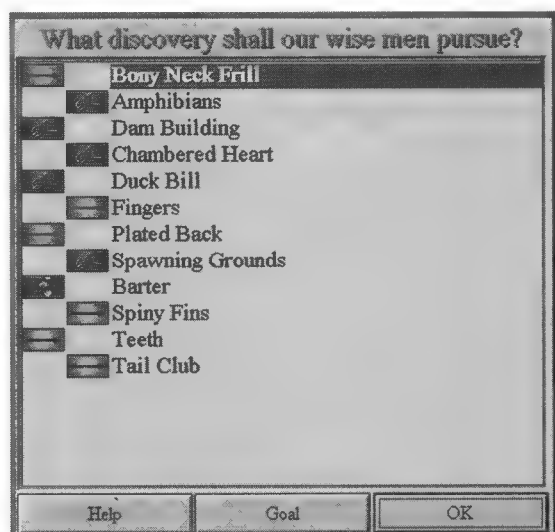
When you decide to make changes to the progression of advances (the tech tree) for your scenario, you're stepping into a deep a can of worms. This is perhaps the most complicated part of *Civilization II*, and therefore the most difficult to modify skillfully. Don't worry. This chapter covers what you'll need to know to sidestep the most common mistakes, whether you want to simply rename things a little bit or completely remake the entire tech tree.

Some Guidelines

In one sense, building a tech tree is like designing units; what you'll want to do depends on your scenario. For historical scenarios, cutting off the advanced lines of research and making minor changes to the existing early advances might be all that's necessary. When you create whole new worlds, you'll almost certainly want to reinvent the entire scope of science. As I did for units, then, I'll leave the joy of creating to you, and offer generalized guidance rather than specific rules.

- ▼ **Make a plan.** It might sound trite, but in the case of advances, it's truly important to have your course mapped out. You can just wade into this process and start thrashing about, but your scenario will suffer for it. Before you touch the tech tree, you should *know* what you intend to do to it—it's the best way to avoid loops, cut-outs, and other nasty problems.
- ▼ **Stay true to your milieu.** This is no less true for being obvious; your scenario will be much more enjoyable if it is internally consistent. That's why no one can research Nuclear Fission in the Alexander the Great scenario, and why Hematite Processing isn't called Iron Working in the Mars scenario (despite the fact that they're virtually the same thing).
- ▼ **Limit the options.** In a scenario, every advance that the player has the prerequisites for is listed when it's time to choose the next research project (see Figure 4-1). To prevent this box from filling up, don't let your tech tree spread out too much as it progresses. Include a few dead ends and branches that rejoin. That way, your tree is longer and doesn't get too thick in the middle.

Figure 4-1:
*The Research
Selection List*



- ▼ **Maintain a balance.** This is easiest if all civilizations have access to the same list of technologies. If you do decide to build partially or completely separate trees, however, do plenty of play-testing to make sure that no one tribe has a distinct advantage.
- ▼ **Put some super stuff near the end.** Research is a time-and-resource consuming enterprise, and your players deserve a reward for all their efforts. Unless the historical realism of your scenario (or some other rationale) precludes it, you should leave some treasure at the end of the science rainbow—powerful units, really useful improvements and wonders, or (naturally) spaceship parts.
- ▼ **Every tribe must be able to reach Future Technology.** Regardless of whether you have one big tree or set up a separate tech tree for each civilization, every tribe must have a route to Future Tech. *Civilization II* crashes whenever there are no more subjects to research—that’s why you can discover Future Tech over and over again. You should never, ever remove Future Tech from the game.

Don’t worry about getting it right the first time. After you’re finished with the tech tree, you should play your scenario *all the way through several times*. That will give you a feel for anything you’ve done wrong, and you can fix it.

If you’d like to know how to create tribe-specific tech trees (like the ones in the Alien Invasion scenario), that information is in Chapter 10: Tricks of the Trade.

TIP

Charting the Tree: One Method

If you’ve ever played *Civilization II*, you’ve consulted the poster. The map of the technology tree that fills the top portion of that poster took a long time (and more than one person) to put together, and it’s still not as clear as we would have liked. The fact is, interesting tech trees are *complicated*. It won’t be as difficult to keep track of your tree as it was to make the poster—you don’t need to print your chart out for other people to understand—but it’s not something you want to leave undone. If you try to keep the tech tree in your head, you’re almost guaranteed to make mistakes. Why bother?

This method is useful when you’re making really substantial changes to the *Civ II* tech tree. If you’re only making slight changes, you can probably keep all the details straight in your mind, and all the mucking about with cards and tape is unnecessary.

NOTE

Here's the method I used when I designed the tech trees for the two scenarios I built for MicroProse. (I created completely new trees for both of those scenarios.) I'm sure it's not the best possible method, but it worked for me. If it doesn't work for you, don't use it. Think of it as a suggestion.

Gather Materials

You'll need to gather a few necessary supplies before you start.

- ▼ **A large board.** First of all, you need a large empty surface that you can write on, erase cleanly, and attach bits of paper to. I used a big erasable marker board (a whiteboard). A chalkboard would work just as well.
- ▼ **Small cards.** Next, you need the bits of paper I mentioned and a way to attach them to the writing surface. I used a handful of business cards and tape. The cards must be large enough for you to write at least three short lines of text on each one.
- ▼ **Paper strips.** You'll also need some smaller paper bits—thinner than the cards you already have, because you're going to tag them to the sides of the others—and a way to attach them to the cards. I used the same tape, and I cut a lot of little strips of paper to stick to the business cards.
- ▼ **Colorful writing utensils.** The last supplies you'll want are writing instruments. For whatever you use to write on the erasable surface, you'll need two or three colors. For writing on the bits of paper, one color will do, but I suggest three.

Organize Advances

The first and most important step in this process is to organize the advances themselves.

- 1 Take your cards and write the name of one advance on each in big letters. Underneath that, save space for the two prerequisites. Don't fill those in yet unless you're using a pencil. You don't want to write in the prerequisites until you're absolutely sure of them; part of the efficacy of this technique is that it enables you to shuffle things around. Keep a few blank cards handy, too.
- 2 Now find a table or other large, flat surface. Lay your advance cards out on it and line them up according to your plan. In the process of trying to organize the cards like this, you'll get a clearer idea of how your tree is structured, and you'll find some of the problems with your tree design.

- 3 When you've figured out the best layout for all the cards, transfer them onto the big board. Leave room in between the cards to write and erase.
- 4 Now, draw lines between the cards, linking each advance to its prerequisites. (This is why it's good to have more than one color to draw with, so that you never confuse lines that cross). This step will bring more problems to your attention. Move the cards around and redraw the lines as much as necessary.
- 5 When you're done, take a step back and look at the overall structure you've created. Look for problems you might have missed while you were busy focusing on the details. For example: Are there too many dead ends? Are any advances unintentionally disconnected? Do some advances seem unnecessary? Are any advances too important? Is the progression unbalanced?
- 6 When you're convinced that the tree is good, go ahead and write in the prerequisites on the cards.

Assign Improvements and Units

The next step is more important than you might think. After you have a solid plan for your tech tree, you can easily figure out what units, city improvements, and wonders you want made available (and obsolete) by what advances. Being able to step back and look at a graphic representation of your whole plan—advances, units, wonders, and improvements all in one picture—is a great tool for helping you balance your scenario. I discuss balancing these parts of the game in other chapters (Step 3: Design Units and Step 5: Make Improvements), so I won't dwell on the details here. This is the procedure.

- 1 Take all those little strips of paper and write down the names of the units, city improvements, and Wonders of the World you plan to include in the scenario—one to a strip. Repeat the process using another color, so that you end up with two strips for each thing, one of each color—one for availability and one for obsolescence.
- 2 Now, attach the strips to the corresponding cards. For example, if an advance makes a unit available, tag the Color 1 strip for that unit to the card for that advance. Find the advance that makes that unit obsolete, and attach the unit's Color 2 strip to that card. (I always tag the first set of strips to the right side of the card and the obsolete strips along the bottom; it helps me to tell them apart at a glance.)

- 3 Once again, step back and look over the whole picture. Don't be surprised if there are dozens of tiny mistakes you need to correct—things becoming obsolete too soon, too many strips attached to the same advance, no strips on an entire portion of the tech tree, and so on. Remember, finding the mistakes and the imbalances is what this process is all about.
- 4 When you finish moving things around, congratulations! You've just finished designing a complete *Civ II* tech tree.

Change the Rules

Now that you've got everything planned out *and* you've already ironed out most of the problems, the rest of the process is a snap. All you need to do is take your cards down off the board and enter the design marked on them into the *Civ II* rules.

Exactly how you do so is the subject of the next section.

Defining Advances

This is the section in which I review the format and content of an advance definition in detail. What's true for terrain and units is just as true for advances—the better you understand the entries in *rules.txt*, the more quickly you'll be able to implement what you imagine.



**FANTASTIC
WORLDS**

If you have *Fantastic Worlds*, you can modify the advances using the Advances Editor on the Editors menu. This method is more convenient and probably easier for most folks, but it does not include any method of displaying the entire interconnected tech tree. You still need to keep track of that yourself.

Understanding the advance definitions is essential. The advances in your scenario are defined in *rules.txt*. Near the beginning of that file is this line:

@CIVILIZE

That's the header marking the beginning of the section containing the definitions of all the advances. Each of the lines of text following it—down to and including the line that begins with **User Def Tech C**—determines the characteristics of a single advance.

If you have *Fantastic Worlds* installed, your *rules.txt* has 7 extra advance slots—
Extra Advance 1 through Extra Advance 7.

FANTASTIC WORLDS

All the lines that start with the @ symbol are header lines. Do not *ever* change *any* of the header lines. If you do, *Civilization II* will probably crash. Also, you should never add lines to or delete lines from *rules.txt*—not even empty lines. In some cases, the program counts lines to find data, and if you remove or add something, you can only cause trouble.

CAUTION



Just like I did for the terrain and unit entries, I'm going to take one of these definition lines apart. The Literacy advance is defined as:

```
Literacy, 5, 2, Wri, CoL, 0, 3 ; Lit
```

Each part of the line is a data field, and each field except the last two must end with a comma. The next to last field must end with a semicolon. Table 4-1 covers the meanings of all the fields.

Table 4-1. Advance Data Fields for Literacy Example

VALUE	FIELD NAME	EFFECT
Literacy,	Name	This is the name of the advance. It shows up in on-screen text. Like the manual says, names shouldn't be more than 15 characters, because they won't fit in all the display boxes and lists. Also, you should use only letters, numbers, and spaces in names; other characters might not be displayed and can occasionally cause program errors.
5,	AI Value	Human players decide for themselves which advances are most important. The computer-controlled civilizations use the number in this field to determine what they should research. The higher this number, the more priority the AI places on getting the advance. This effect is modified by the value of the next field, Civilized Modifier. You should not set this field to zero. The highest value in the original game is 8, but higher values do not cause problems. (There must certainly be a maximum possible value, but no scenario reasonably needs to use values above 9.)

2	Civilized Modifier	The priority the AI places on an advance depends on the AI Value field, but only after it is modified by the value in this field. When you determine the personality of each tribe, one of the characteristics you set is the relative militarism (for details, refer to Step 7: “Scenario” Means “Situation”). This field plays on that setting. For a “civilized” tribe, a positive value here adds to the priority of this advance, and a negative value detracts from the value. The opposite holds true for “militaristic” tribes. The highest value in the original game is 2 and the lowest is -2, but higher and lower values do not cause problems. (There must certainly be maximum and minimum possible values, but no scenario reasonably needs to use values beyond 9 and -9.)
Wri	Prerequisite 1	For a civilization to be able to research this advance, it must first discover both prerequisite advances. This is the first prerequisite advance—denoted here by the advance’s abbreviation. (Other than <i>Civilization II</i> crashing, this is why you should never change the abbreviation of any advance.) <i>Nil</i> in this field or both prerequisite fields means that the advance has no prerequisites, thereby allowing every civilization to research it from the beginning of the scenario. <i>No</i> in both prerequisite fields prevents the advance from appearing in the scenario.
CoL	Prerequisite 2	For a civilization to be able to research this advance, it must first discover both prerequisite advances. This is the second prerequisite advance—denoted here by the advance’s abbreviation. <i>Nil</i> in this field means that the advance has only one prerequisite (the first one). <i>Nil</i> in both prerequisite fields means that the advance has no prerequisites, thereby allowing every civilization to research it from the beginning of the scenario. <i>No</i> in both prerequisite fields prevents the advance from appearing in the scenario.
0	Epoch	This field and the next one, Category, control which of the advance icons is associated with this advance. These icons (as explained later in this chapter) are organized by rows and columns. The Epoch field determines which column the icon for this advance is drawn from. There are four columns, numbered 0 through 3.

Along with the previous field, Epoch, this field controls which of the advance icons is associated with this advance. The icons (as explained later in this chapter) are organized by rows and columns. The Category field determines which row the icon for this advance is drawn from. There are five rows, numbered 0 through 4. Note that this field must end with a semi-colon, not a comma.

This is the official abbreviation for this advance slot. *Do not change this.* It doesn't matter what the name of the advance is; this is always the abbreviation for the advance defined on this line.

Changing the Icons

Every advance has an icon associated with it. You see it in the advance selection list. If you've read table 4-1 already, you know that the Epoch and Category of the advance control which icon this is. If you decide that it's appropriate for your scenario to have different icons than the original game has, you can go right ahead and change them. To do so, you need a way to edit GIF files. (For more details on GIF editors, see Step 1: Getting Started.)

If you have *Fantastic Worlds*, you can modify the advance icons using the Icon Editor in the Advances Editor on the Editors menu. This method is more convenient and easier than a GIF editor for most folks, but keep in mind that there are only sixteen icons. When you edit the icon for one advance, it affects all the other advances with the same Epoch and Category.

**FANTASTIC
WORLDS**

The Table Method

For those of you who are experienced Microsoft Word users, there's a shortcut method you can use when redesigning advances for your scenarios. If you convert the advance definition portion of *rules.txt* into a 7-column table, it makes it much easier to keep track of which field is which. Here's one way to do it (not necessarily the best):

- 1 Create a new Word document, and copy the text of the advance definition lines into the document.
- 2 Use the Replace feature to eliminate all the spaces.
- 3 Convert the text to a table, using the comma as the field separator. You should end up with 7 columns. The last column includes both the Category and the abbreviation, but since you can't change the abbreviations, it shouldn't be a problem.
- 4 Put borders around the cells (if you want to—it helps me).

Now, do all your editing. When you're done and ready to put things back, reverse the process:

- 1 Select the whole table and convert it back to text, separating the text with commas.
- 2 Select all the text and use the Copy feature to put that text into the Windows Clipboard.
- 3 Open *rules.txt* and find the `@CIVILIZE` line.
- 4 Select all the old advance definition lines, then use the Paste feature to replace them with your edited text.
- 5 Save *rules.txt* as a Text Only file. (If it appears, don't believe the prompt about "formatting that cannot be saved in text format"—it's a crock. There's no non-text formatting necessary in *rules.txt*.)

There's a similar shortcut that could help experienced Microsoft Excel users, but it's more complicated. If neither of these is appropriate for you, don't worry. Editing the original text file is not really difficult; these shortcuts are just alternate ways of approaching the task.

The remainder of this section assumes that you have a GIF editor or *Fantastic Worlds*.

All sixteen of the advance icons (plus a lot of other things that aren't relevant right now) are contained in the file *icons.gif*. Once you understand the way this file is organized and how to find the icon you're after, you should have no trouble editing these icons.

The Palette and Purple Lines

Files in GIF format are one example of what are called *paletted* graphics files. The palette (sometimes called a “color table”) is a collection of colors defined for use in the file, and no other colors appear in that file. Most graphics programs include a way to display the palette, usually as a box full of colors you can pick from. All of the files for *Civilization II* use the same palette, and you should never change the palette in any of the files—ugly graphic problems result, and the game could crash, too.

The *Civ II* palette contains some colors that are recognized by the program as “not to be displayed” colors. These are mostly “transparent” colors that we can use in the files to help us position graphic elements correctly. All of the advance icons are surrounded by a rectangular outline of purplish lines (see Figure 4-2). That purplish color is a transparent color. When the icon is displayed on the screen, the purplish lines are not visible.



Figure 4-2: Advance icons are in purple boxes.



CAUTION

Some colors in the palette are very similar to the transparent colors, especially the transparent shade of gray. If you get them mixed up, your game will look pretty bad, but probably won't crash. Just re-edit the file and replace the wrong color with the right one, and the problem goes away.

Those purple boxes are used by the program as reference to tell where the icons are. Every icon is in a box of exactly the right size and shape, and you should never mess with the lines. Work inside the lines.



Never move or erase any of the purple lines. Don't add new ones, either.

CAUTION



Columns and Rows

If you look at *icons.gif*, you'll probably recognize the advance icons (and several other graphics). Anything you see that does not appear in the game is in an *inactive area* of the GIF file (see Figure 4-3). Anything outside of the *active areas* is either ignored by the game program or is something you should never touch, so there's no point in changing any of it. The exact location of the advance icons is noted in Figure 4-4.

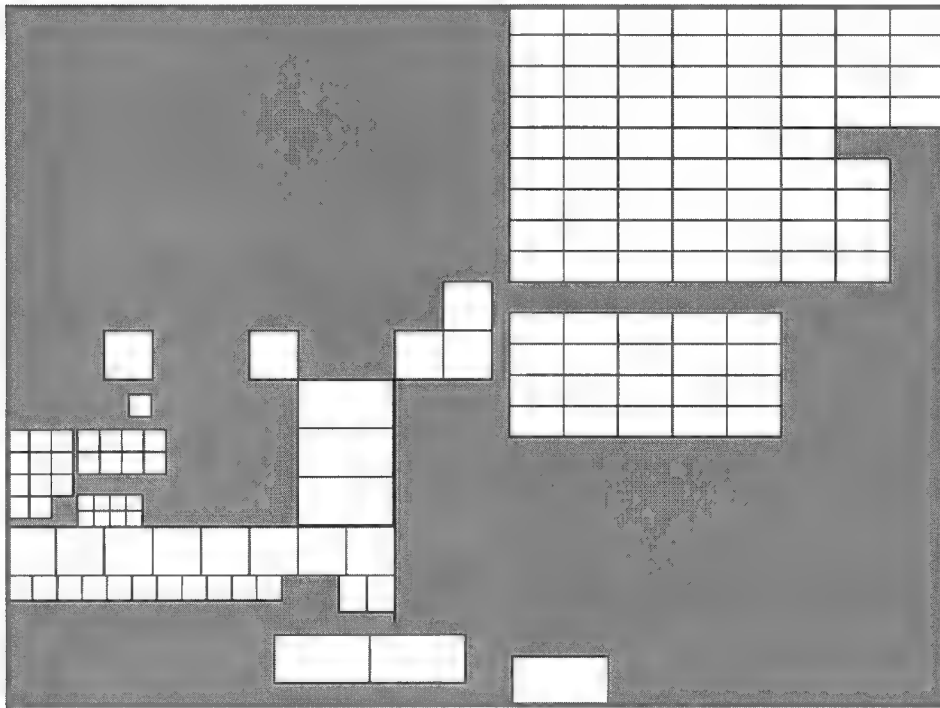
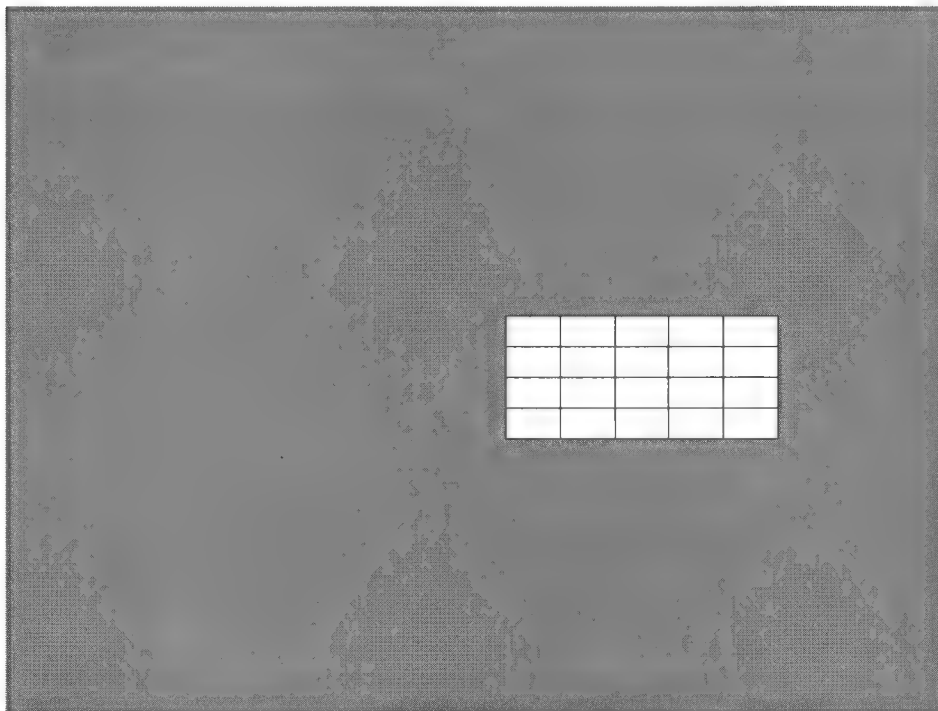


Figure 4-3: Active areas of *icons.gif*

Figure 4-4:
Where the advance
icons are



Each row of icons corresponds to a single Epoch, from 0 at the top down to 3 at the bottom. Each column represents a Category, from 0 at the left to 4 at the far right. The combination of Epoch and Category determines, for each advance, which icon is associated with that advance. Thus, for example, the icon in row 2, column 2 appears in the game for all advances with Epoch=1 (Renaissance) and Category=1 (Economic). (Note that you can change the names of the epochs and categories by editing the file *labels.txt*. For more detail, see Step 6: The Foundations of Civilization.)

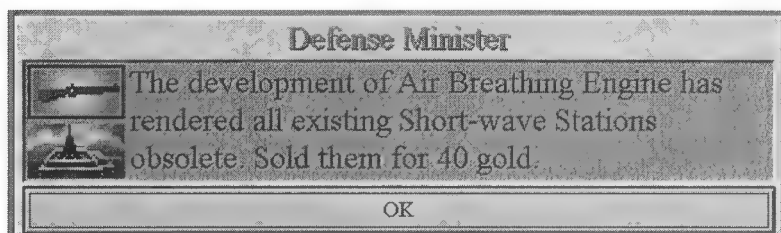
Do Not Touch

Here's a general rule that I apply to all the *Civ II* files: If you don't know what it is and how it works, don't touch it.

When it comes to the icons file, there is only one part that neither you nor I nor anyone else should change: the one we've already mentioned—the purple lines.

Fixed Side-Effects

I've been discussing the characteristics of advances that you *can* change, but it's also important to understand what you *can't* change. The features that seem to cause the most confusion and annoyance for scenario designers are the special effects certain advances have. There's no way to change these effects, they're *hard-coded*—determined by the program code, not by anything in *rules.txt*. The best way to avoid having these features cause problems and unexpected occurrences in your scenarios is to know about them ahead of time. That way, you can eliminate them from your scenario or use them to your advantage.



The most common special effect is that certain advances make new forms of government available. It's obvious to any veteran *Civ II* player which ones those are. Table 4-2 lists the others—all the rest of the advances that have hard-coded side effects, along with brief descriptions of those effects.

Table 4-2. Fixed Effects of Advances

SLOT	ORIGINAL NAME	FIXED EFFECT
5	Automobile	In combination with Electronics (24), changes city icons to the <i>modern</i> style (row 6 in <i>cities.gif</i>).
7	Bridge Building	Allows units with a Role of 5 (settler-types) to build roads on terrain squares with rivers.
15	Communism	Reduces the effectiveness of the improvement in slot 11 (Cathedral) by one citizen.
18	Construction	Allows units with a Role of 5 (settler-types) to build Fortresses.
24	Electronics	Increases the effectiveness of the improvement in slot 14 (Colosseum) by one citizen. In combination with Automobile, changes city icons to the <i>modern</i> style (row 6 in <i>cities.gif</i>).

32	Fusion Power	Eliminates the risk that civil disorder will produce a nuclear meltdown in cities with the improvement in slot 21 (Nuclear Plant).
35	Gunpowder	Makes all of the discovering civilization's Barracks obsolete; they're sold off immediately.
37	Industrialization	Changes city icons to the <i>industrial</i> style (row 5 in <i>cities.gif</i>).
53	Mobile Warfare	Makes all of the discovering civilization's Barracks obsolete; they're sold off immediately.
55	Monotheism	If the Cathedral improvement (or the renamed equivalent) is not tied to this advance, that improvement has no effect.
56	Mysticism	Doubles the effect of the improvement in slot 4 (Temple).
57	Navigation	Decreases the chance that a sea unit with the <i>Might be lost away from land</i> attribute will disappear in deep water.
59	Nuclear Power	Adds one to the movement allowance of all the discovering civilization's sea units (those with a Domain of 2).
60	Philosophy	Immediately confers the next advance.
66	Radio	Allows units with a Role of 5 (settler-types) to build Airbases.
67	Railroad	Allows units with a Role of 5 (settler-types) to build Railroads.
70	Refrigeration	Allows units with a Role of 5 (settler-types) to double-irrigate terrain to create Farmland.
75	Seafaring	Decreases the chance that a sea unit with the <i>Might be lost away from land</i> attribute will disappear in deep water.
82	Theology	Increases the effectiveness of the improvement in slot 11 (Cathedral) by one citizen.
89	Future Tech	Adds 5 points to the Civilization Score.

These can be annoying, but once you learn to work around them and with them, it's safe to say you're well on your way to mastering advances. Working with the fixed effects also helps prepare you for the next step—City Improvements and Wonders of the World—because *all* of their effects are fixed. ∞

STEP 5

MAKE IMPROVEMENTS

NOW THAT YOUR MAP is essentially complete, your units are designed, and you've charted the course of research, let's turn to city management. Specifically, this chapter deals with what you can and can't do with the City Improvements and Wonders of the World.

It's helpful, but not really necessary, to have your advances mapped out before you create the units for your scenario. When you begin making City Improvements and Wonders of the World, however, it's much more important to have a completed tech tree in hand. Even though you can't change much about them, it's surprisingly easy to throw off the game balance of a scenario by messing with the improvements and wonders.

A Few Guidelines

The main problem is that these buildings are *powerful*. I don't mean just the wonders, either. While it's true that wonders have extreme effects that can change the course of the game, they are limited because each one is unique—it can only be built once. The effects of improvements are smaller in scope, but multiplied several times over—every civilization can build them in almost every city—they easily match the wonders in terms of the pressure they exert on the balance of the game.

Every scenario is different (that's the fun of it), so I can't really advise you specifically what to do and what not to do. However, through trial and error (*lots* of error), the scenario designers at MicroProse have developed a few guidelines that seem to help keep things from getting too unbalanced.

- 1 If all the civilizations have equal access to an improvement—everyone can get the necessary advance and build it—you generate no imbalance.
- 2 It's usually a bad idea to remove all the crutches of one type—money, production, food, defense, or happiness. Taking out the Library, for example, also toasts the University and Research Lab; then science becomes *much* less important to the player, because it's nearly impossible.
- 3 Removing improvements and wonders should be limited to those that are inappropriate to the theme of the scenario or the historical period—and can't be renamed to fit.
- 4 If you set the upkeep for the money crutches (Marketplace, Bank, and Stock Exchange) too high, they become useless in small cities—they might barely even pay for themselves. Set them too low, however, and they can be overpoweringly profitable.
- 5 Reordering when the improvements become available is fine—as long as your schedule is designed to provide what cities are likely to need—a food boost, extra income, or whatever—at about the time they'll need it. Foresight helps with this, but there's no substitute for playtesting.
- 6 Wonders and Barracks are a special case, because their effects expire. Schedule each of these so that there's time to build it *and* enjoy the effects for quite a few turns (20 at the *very* least) before obsolescence rips it away. If a thing is not tempting, no one will build it, and then what's the point?

Just keep in mind that the original game is a complex balancing act, brought about only through untold man-years of analysis, playtesting, and fiddling. Any time you mess with something, be mindful of the long-term effects.

Now, assuming we're all dedicated to conserving the balance of the game in our scenarios, let's get on with the nuts and bolts of it.

Changing Things

While there's not much you can change about improvements, there are a few things you can do to City Improvements and Wonders of the World. You can:

- ▼ Rename them.
- ▼ Remove some or all of them from the scenario.
- ▼ Change the advance that allows each to be built, including making some available from the start.
- ▼ Raise or lower (but not eliminate) the costs to build them.
- ▼ Change the sounds associated with the completion of certain buildings.
- ▼ Edit the icon that represents each one.
- ▼ Change or eliminate the maintenance cost of improvements.
- ▼ Have wonders expire at different times—or not at all.

As usual, most of the work takes place in the file *rules.txt*, so that's where we'll start.

The Rules

This section should seem familiar to you from previous chapters. In it, I review the format and content of a portion of the definitions in *rules.txt*. The difference is that when it comes to City Improvements and Wonders of the World, there are two sections of that file you can edit—the definitions and the obsolescence schedule for the Wonders.

FANTASTIC WORLDS

If you have *Fantastic Worlds*, you can modify both improvements and wonders using the City Improvements Editor on the Editors menu. This method is more convenient and probably easier for most folks, and it presents all the information in one place.

Improvement Entries

We'll start with the definition entries. These include both City Improvements and Wonders of the World in one section. Search through *rules.txt* until you find this line:

```
@IMPROVE
```

That's the header marking the beginning of the section. Each of the lines of text following it—down to and including the line that begins with **Cure for Cancer**—determines the characteristics of a single improvement.

CAUTION

All the lines that start with the @ symbol are header lines. Do not *ever* change *any* of the header lines. If you do, *Civilization II* will probably crash. Also, you should never add lines to or delete lines from *rules.txt*—not even empty lines. In some cases, the program counts lines to find data, and if you remove or add something, you can only cause trouble.

TIP

Take note of the first line of the improvement definitions—**Nothing, 1, 0, nil,**—which is absolutely necessary and shouldn't be renamed. This defines the state of a city with no improvements at all. You can use this to give cities themselves an upkeep cost, but you shouldn't change anything else on this line.

Just as we've done for the other types of entries, let's take one of these definition lines apart. The Recycling Center is defined as:

```
Recycling Center, 20, 2, Rec,
```



Each part of the line is a data field, and each field including the last one must end with a comma. Table 5-1 covers the meanings of all the fields.

Table 5-1. Data Fields for Recycling Center Example

VALUE	FIELD NAME	EFFECT
Recycling Center	Name	This is the name of the improvement. It shows up in plenty of on-screen text, but has no other effects. Like the manual says, names shouldn't be more than 15 characters, because they won't fit in all the display boxes and lists. Also, you should use only letters, numbers, and spaces in names; other characters might not be displayed and can occasionally cause program errors.
20	Cost	This sets the cost in shields to build the improvement. As with the other cost fields, the number in this field is multiplied by the <i>Number of Rows in Shield Box</i> constant to get the actual cost of the building. (Thus, based on the default constant—10 rows—building the Recycling Center requires 200 shields.) Make sure that this cost reflects how useful and important the improvement is in your world. This field should never be set to zero.
2	Upkeep	In this field, you determine the per turn maintenance cost (in gold) for the improvement. When this cost is not paid, improvements are generally sold off automatically. If this is set to zero, the improvement has no upkeep cost. Note that you <i>can</i> give an upkeep cost to a wonder of the world, but it is ignored by the program and will have no effect on the game (it won't be paid).
Rec	Prerequisite	A civilization must discover this advance—denoted here by the advance's abbreviation—before it can begin to build the improvement. (This is another reason why you should never change the official abbreviations of the advances.) <i>Nil</i> in this field means that the improvement has no prerequisite, thereby allowing every civilization to build it from the beginning of the scenario. <i>No</i> here prevents the improvement from appearing in the scenario.

When you set the prerequisite advance for an improvement, double-check your tech tree to make sure that that advance is actually in your scenario.

CAUTION



Wonder Enders

Every Wonder of the World can be made obsolete—that is, its effects cease—by the discovery of a particular advance. Remember that when *any* civilization discovers the advance, the effects of the wonder are terminated.

Search through *rules.txt* until you find this line:

```
@ENDWONDER
```

That’s the header marking the beginning of the section. Each of the lines of text following it—down to and including the line `Fl i , ; Cure for Cancer` determines the expiration advance for a single wonder. These lines are one example of data that is clearly found and used by the program according to its position. There is no “Name” field that would allow the program to reference lines by the name of the wonder. The only active field on the line is the name of the obsolescence advance.

Let’s take one apart. The Colossus’s expiration is defined as:

```
Fl i , ; Colossus
```



The two parts of the line are data fields. The first field must end with a comma; the second must begin with a semicolon. Table 5-2 explains the meanings of the fields.

Table 5-2. Obsolescence Fields for Colossus Example

VALUE	FIELD NAME	EFFECT
Fl i ,	Advance	This is the abbreviation of the advance the discovery of which makes the wonder obsolete. (This is yet another reason why you should never change the official abbreviations of the advances.) Remember that civilizations can still build obsolete wonders, they simply have no effects except on the score. <i>Nil</i> in this field allows the wonder to function until the end of the scenario.
; Colossus	Label	This field is just a notation of which wonder is made obsolete by the advance listed on this line. You can change this label all you want—it has no effect on the game—but you should never remove the semicolon.

When you set the obsolescence advance for a wonder, double-check your tech tree to make sure that that advance is actually in your scenario.

CAUTION



Modifying Icons

Since there is so little you can do to the functional definitions of the improvements and wonders, it's quite tempting (perhaps less so in historical scenarios) to make them look different. It's not difficult, but you need a way to edit GIF files.

If you have *Fantastic Worlds*, you can modify the looks of every improvement and wonder icon using the Icon Editor in the City Improvements Editor on the Editors menu. This method is more convenient and easier than a GIF editor for most folks. (For more details on GIF editors, see Step 1: Getting Started.)

FANTASTIC
WORLDS



The remainder of this section assumes that you have a GIF editor or *Fantastic Worlds*.

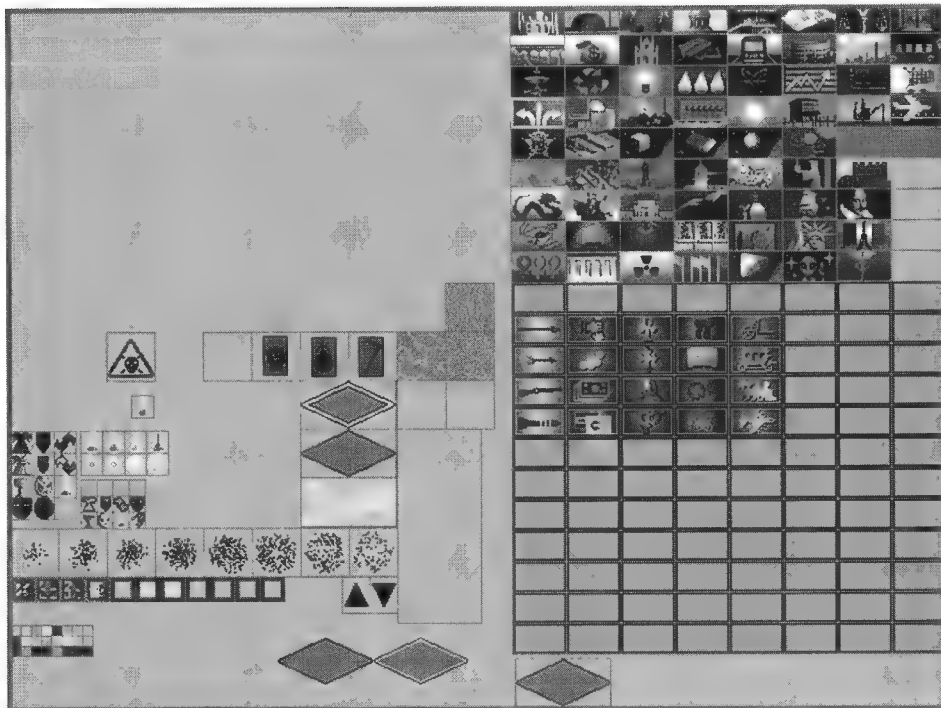
All the improvement and wonder icons (plus a lot of other things that aren't relevant right now) are contained in the file *icons.gif*. Once you understand the way this file is organized and how to find the icon you're after, you should have no trouble editing the looks of improvements and wonders.

The Palette and Purple Lines

Files in GIF format are one example of what are called *paletted* graphics files. The palette (sometimes called a “color table”) is a collection of colors defined for use in the file, and no other colors appear in that file. Most graphics programs include a way to display the palette, usually as a box full of colors you can pick from. All of the files for *Civilization II* use the same palette, and you should never change the palette in any of the files—ugly graphic problems result, and the game could crash, too.

The *Civ II* palette contains some colors that are recognized by the program as “not to be displayed” colors. These are mostly “transparent” colors that we can use in the files to help us position graphic elements correctly. All of the improvement icons are surrounded by a rectangular outline of purplish lines (see Figure 5-1). That purplish color is a transparent color. When the improvement icon is displayed on the screen, the purplish lines are not visible.

Figure 5-1:
Improvement icons
are in purple boxes.



Those purple boxes are used by the program as reference to tell where the icons are. Every icon is in a box of exactly the right size and shape, and you should never mess with the lines. Work inside the lines.



CAUTION

Never move or erase any of the purple lines. Don't add new ones, either. Also, some colors in the palette are very similar to the transparent colors, especially the transparent shade of gray. If you get them mixed up, your game will look pretty bad, but probably won't crash. Just re-edit the file and replace the wrong color with the right one, and the problem goes away.

Room for Improvement

If you look at *icons.gif*, you'll probably recognize the improvement icons and also notice several other graphics. Anything you see that does not appear in the game is in an *inactive area* of the GIF file (see Figure 5-2). Anything outside of the *active areas* is either ignored by the game program or is something you should never touch, so there's no point in changing any of it.

Each icon corresponds to a single improvement or wonder entry in *rules.txt*. The position of the icon determines which improvement it represents (see Figure 5-3). When you rename improvements, keep track of the original name of each, so that when you begin modifying the icons, you know exactly which icon to change for each of your new improvements.

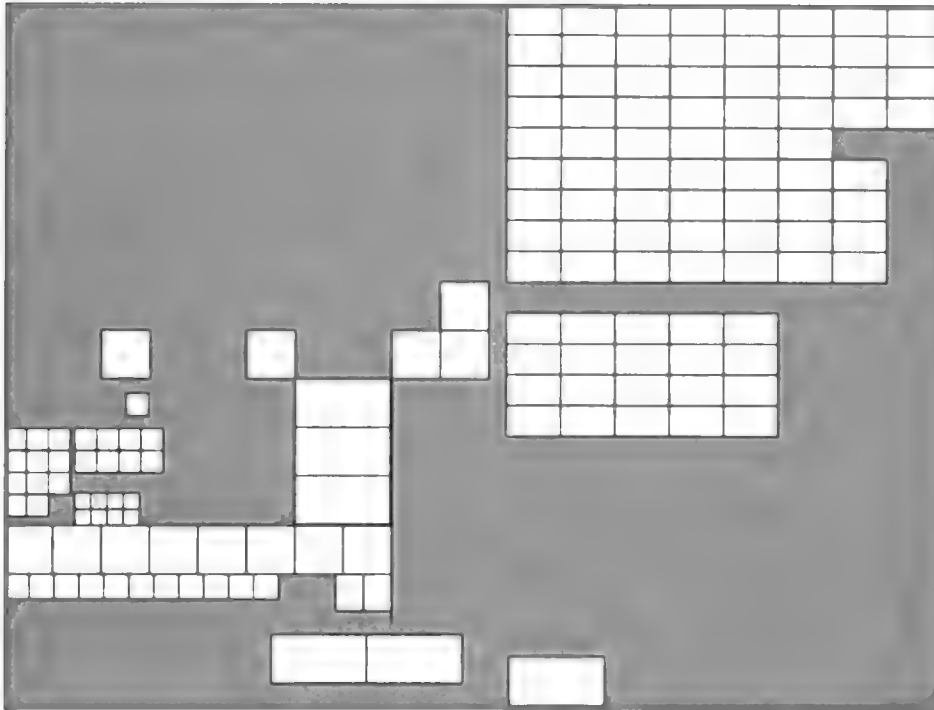


Figure 5-2: Active areas of Icons.gif

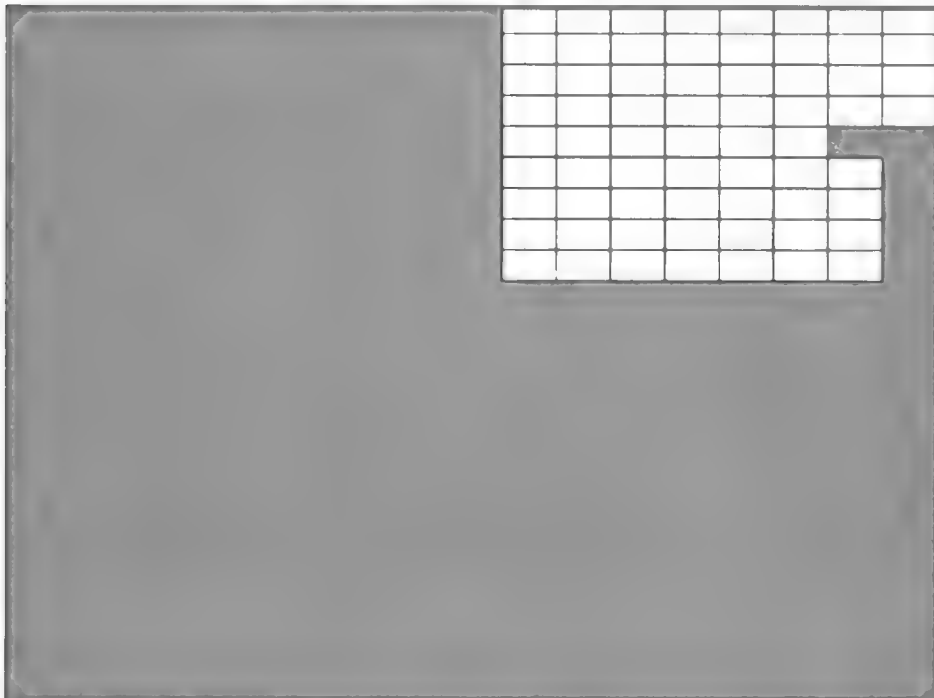


Figure 5-3: Where the improvements are

Do Not Touch

Here's a general rule that I apply to all the *Civ II* files: If you don't know what it is and how it works, don't touch it.

When it comes to the icons file, there is only one part that neither you nor I nor anyone else should change: the one we've already mentioned—the purple lines.

Replacing Sounds

When you're changing the improvements around, there's a natural temptation to go the whole route. If you've customized the definition and the pictures, why not the sound effects, too? It can be especially tempting in this case, since there's so little you can really change when it comes to improvements and wonders.

FANTASTIC WORLDS

Fantastic Worlds does *not* offer you any shortcuts for changing the sound associated with improvements and wonders, so pay attention!

First of all, let's establish what sound is played when each City Improvement is completed. In most cases, the program chooses one of three sound files at random. Those three files are *cheers1.wav*, *cheers2.wav*, and *cheers3.wav*. This is inconvenient for scenario builders, because it is not possible to associate a particular sound with the majority of the city improvements. (Even in the Age of Reptiles scenario, which has plenty of new sounds, we didn't bother with these.) It can be fun to replace some or all of the cheers files, but whatever sound you insert must be appropriate to just about all of the improvements that can be built in your scenario.

The situation is similar, but perhaps a little worse, for Wonders of the World. They *all* share the same sound—*newonder.wav*. Again, whatever sound you use must befit every wonder that can be built in your scenario.

NOTE

Notice that, in both cases, I said that the sounds must be appropriate for the buildings that can be built in your scenario. You do not need to consider any improvements or wonders that: (1) you have eliminated, (2) are already completed when the scenario begins, or (3) can be built only by unplayed civilizations (those that you do not expect humans to play—what we call *inactive* civilizations).

Table 5-3 lists the nine City Improvements that have distinct sounds associated with them. These nine do *not* use the cheers files.

Table 5-3. Custom Improvement Sounds

IMPROVEMENT	SOUND FILE
Aqueduct	<i>Aqueduct.wav</i>
Bank	<i>Newbank.wav</i>
Barracks	<i>Barracks.wav</i>
Cathedral	<i>Cathedrl.wav</i>
Marketplace	<i>Mrktplce.wav</i>
Stock Exchange	<i>Stkmarkt.wav</i>
SSComponent	<i>Bldspcsh.wav</i>
SSModule	<i>Bldspcsh.wav</i>
SSStructural	<i>Bldspcsh.wav</i>

To change these sounds in your scenario, you simply insert a sound file of your own—with the exact same name as the file you want to replace—into the Sound subfolder. When the time comes to play the sound, the program searches there for the file with the appropriate name, then plays it.

The only catch is that *Civ II* sounds need to be in a specific format. As mentioned before, in case you're skipping around the book, here it is again.

For *Civilization II* to be able to play a WAV file, it should be in this format:

- ▼ **22kHz**—This is the only sampling rate that's guaranteed to work. If you try to use anything else, you're on your own.
- ▼ **8-bit**—The game works well with 16-bit sounds, but that's *only if* you're not using a pre-95 version of Windows *or* you have a 16-bit sound card. Versions of Windows earlier than 95 (3.1x) cannot play 16-bit sounds on an 8-bit sound card.
- ▼ **Mono**—Once again, if you have Windows 95 (or later) or a 16-bit sound card, you can probably use stereo sound effects without a problem. Otherwise, for get it.

Why all the strictures and the low quality sound? Remember, *Civilization II* was developed to work with Windows 3.1 (and anything newer, of course). Sound capabilities have advanced quite a bit in the last couple of years.

Effects Reference

When you're modifying the definitions of the City Improvements and Wonders of the World, it helps to have a quick reference to their various effects. That's exactly what this is. For ease of use, the entries are ordered exactly as they are in *rules.txt*.

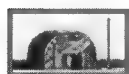
Palace



The effects of this building are as follows:

- ▼ The building defines the civilization's capital city.
- ▼ It prevents corruption and waste in the capital, and corruption and waste in other cities is in proportion to their distance from the Palace. (These also depend on the type of government.) If there is no Palace, all cities are considered to be at a distance of 32.
- ▼ Capture of this city destroys the civilization's spaceship (under construction or in transit).
- ▼ When built, it causes any existing Palace to disappear.
- ▼ This building cannot be sold.
- ▼ Upkeep is not charged until a second Palace is built (the capital is moved).

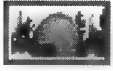
Barracks



The effects of this building are as follows:

- ▼ The city produces veteran ground units (those with a Domain of 0).
- ▼ Any ground unit spending a full turn in the city returns to full combat strength (is repaired completely).
- ▼ The building is automatically sold off when the civilization discovers the advance in slot 35 (Gunpowder).
- ▼ The building is automatically sold off when the civilization discovers the advance in slot 53 (Mobile Warfare).

Granary



A Granary has only one effect. When the Food Storage Box is filled and the city experiences a rise in population, the box is emptied halfway, rather than becoming completely empty, as happens when there is no Granary.

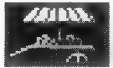
Temple



The effects of this building are as follows:

- ▼ It makes 1 unhappy citizen in the city content.
- ▼ The discovery of the advance in slot 56 (Mysticism) increases the number of citizens affected by one.

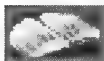
MarketPlace



The effects of this improvement are as follows:

- ▼ It increases the tax and luxury output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Bank or a Stock Exchange (but not both) in the city, the tax and luxury output becomes 200 percent of base.
- ▼ If there are both a Bank and a Stock Exchange, the three together make the tax and luxury output 250 percent of base.
- ▼ If there is no Marketplace in a city, that city cannot build a Bank. However, if the Marketplace is destroyed or sold after the Bank is built, the Bank does not disappear.

Library



The effects of this building are as follows:

- ▼ It increases the science output of the city by 50 percent of its base value.
- ▼ If there is a University or a Research Lab (but not both) in the city, the science output becomes 200 percent of base.
- ▼ If there are both a University and a Research Lab, the three together make the science output 250 percent of base.
- ▼ If there is no Library in a city, that city cannot build a University. However, if the Library is destroyed or sold after the University is built, the University does not disappear.

Courthouse



The effects of this building are as follows:

- ▼ It halves Waste and Corruption in the city. This effect is not relevant under Communism or Democracy (or their renamed equivalents), because in *Civilization II* those forms of government suffer neither Corruption nor Waste.
- ▼ It increases the price for an enemy diplomatic unit to incite a revolt in the city. This effect is not relevant under Democracy (or its renamed equivalent).
- ▼ Under a Democracy (or its renamed equivalent), the building makes 1 citizen in the city happy—a content citizen if there is one, an unhappy one if there is not.



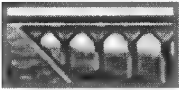
City Walls



The effects of this building are as follows:

- ▼ All units in the city defend at 3 times their normal Defense Factor, except against units with the *Ignores City Walls* attribute. (The details on unit attributes are in Step 3: Design Units.)
- ▼ It prevents the loss of population normally caused by any combat loss by a unit in the city (i.e.: a successful attack that destroys that unit).

Aqueduct



The effects of this building are as follows:

- ▼ It enables the city's population to grow beyond the size specified in the Cosmic Principles. (For more details, see Step 6: The Foundations of Civilization.) The default size limit is 8.
- ▼ If there is no Aqueduct in a city, that city cannot build a Sewer System.
- ▼ If the Aqueduct is destroyed or sold after the city has surpassed size 8, it has no effect on the city—unless the city later shrinks below size 8, in which case an Aqueduct is needed for the city to grow past 8 again.

Bank

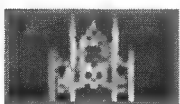


The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Marketplace (or the renamed equivalent) in the city. The Bank does not, however, disappear if the Marketplace is later destroyed or sold.
- ▼ It increases the tax and luxury output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Marketplace or a Stock Exchange (but not both) in the city, the tax and luxury output becomes 200 percent of base.

- ▼ If there are both a Marketplace and a Stock Exchange, the three together make the tax and luxury output 250 percent of base.
- ▼ If there is no Bank in a city, that city cannot build a Stock Exchange. If the Bank is destroyed or sold after the Stock Exchange is built, the Stock Exchange does not disappear.

Cathedral



The effects of this building are as follows:

- ▼ It makes 3 unhappy citizens in the city content.
- ▼ If this city improvement does not have the advance in slot 55 (Monotheism) as its Prerequisite, it has no happiness effect at all.
- ▼ The discovery of the advance in slot 15 (Communism) reduces the number of citizens affected by one.
- ▼ The discovery of the advance in slot 82 (Theology) increases the number of citizens affected by one.

University

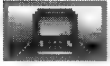


The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Library (or the renamed equivalent) in the city. The University does not, however, disappear if the Library is later destroyed or sold.
- ▼ It increases the science output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Library or a Research Lab (but not both) in the city, the science output becomes 200 percent of base.
- ▼ If there are both a Library and a Research Lab, the three together make the science output 250 percent of base.

- ▼ If there is no University in a city, that city cannot build a Research Lab. If the University is destroyed or sold after the Research Lab is built, the Research Lab does not disappear.

Mass Transit



Mass Transit has only one effect. This improvement completely eliminates the pollution potential caused by population in the city—even after the civilization has discovered the advance in slot 5 (Automobile). This does not affect the pollution potential from other causes. (Other considerations that affect the probability of a city causing pollution include the Difficulty Level, nuclear explosions, and high shield production.)

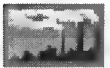
Colosseum



The effects of this building are as follows:

- ▼ It makes 3 unhappy citizens in the city content.
- ▼ The discovery of the advance in slot 24 (Electronics) increases the number of citizens affected by one.

Factory

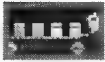


The effects of this building are as follows:

- ▼ It increases the production shield output of the city by 50 percent of its base value—to 150 percent (rounded down).
- ▼ If there is no Factory in a city, that city cannot build any of the five “plant” improvements: Manufacturing Plant, Hydro Plant, Nuclear Plant, Power Plant, and Solar Plant. However, if the Factory is destroyed or sold after one or more plants are built, none of the plants disappear.
- ▼ If there is one plant in the city, the combination of the Factory and the plant raises the production shield output to 200 percent of its base value. If the Factory is later destroyed or sold, the 50 percent increase due to the plant continues.

- ▼ If there are two plants in the city, the combination of the Factory and the plants raises the production shield output to 250 percent of its base value. No more than two plants can contribute to the production shield output of a city at one time. If the Factory is later destroyed or sold, the 100 percent increase due to the plants continues.
- ▼ Note that high shield production is a risk factor for pollution. Other considerations that affect the probability of a city causing pollution include the Difficulty Level, nuclear explosions, population, and whether or not the civilization has discovered the advance in slot 5 (Automobile).

Manufacturing Plant

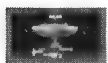


The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Factory (or the renamed equivalent) in the city. The Manufacturing Plant does not, however, disappear if the Factory is later destroyed or sold.
- ▼ It increases the production shield output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Factory or another plant (Hydro, Solar, Nuclear, or Power) in the city, the combination of that and the Manufacturing Plant raises the production shield output to 200 percent of its base value.
- ▼ If there are both a Factory and another plant, the combination of the three raises the production shield output to 250 percent of its base value. (Note that no more than one of the other types of plant can contribute to a city's production at one time.)
- ▼ Note that high shield production is a risk factor for pollution. Other considerations that affect the probability of a city causing pollution include the Difficulty Level, nuclear explosions, population, and whether or not the civilization has discovered the advance in slot 5 (Automobile).



SDI Defense



The effects of this improvement are as follows:

- ▼ It prevents nuclear units from affecting the city and anything within 3 squares of it. A nuclear unit that attacks is still destroyed. However, note that SDI Defense does not prevent an undefended city from being captured by a nuclear ground unit.
- ▼ All units in the city add 50 percent (rounded down) to their Defense Factor when defending against missiles. (In combination with the SAM Missile Battery, this results in a 250 percent defense against missiles.)
- ▼ This improvement does not prevent the effects of nuclear blasts caused by Spy units or exploding Nuclear Plants.

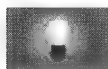
Recycling Center



The effects of this building are as follows:

- ▼ It cuts the pollution potential caused by shield production in the city by two-thirds (down to 33 percent). This does not affect pollution potential from other causes.
- ▼ It negates the pollution reducing effects of any Hydro Plant or Nuclear Plant in the city. Note that this does *not* affect any of the plants' other effects.

Power Plant

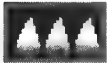


The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Factory (or the renamed equivalent) in the city. The Power Plant does not, however, disappear if the Factory is later destroyed or sold.
- ▼ It cannot be built if there is a more advanced plant (Solar, Hydro, or Nuclear) already in the city. Building one of those does not cause the Power Plant to disappear, but does negate its effects.
- ▼ It increases the production shield output of the city by 50 percent of its base value (rounded down).

- ▼ If there is a Factory or Manufacturing Plant in the city, the combination of the Power Plant and that improvement raises the production shield output to 200 percent of its base value.
- ▼ If there are both a Factory and a Manufacturing Plant in the city, the combination of the three raises the production shield output to 250 percent of its base value.

Hydro Plant



The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Factory (or the renamed equivalent) in the city. The Hydro Plant does not, however, disappear if the Factory is later destroyed or sold.
- ▼ It cannot be built unless there is at least one Mountain or River terrain square directly adjacent to the City Square—or the City Square itself is one of these.
- ▼ It cannot be built if there is a more advanced plant (Solar or Nuclear) already in the city. Building one of those does not cause the Hydro Plant to disappear, but does negate its effects.
- ▼ If there is a Power Plant already in the city, the Hydro Plant negates its effects.
- ▼ It increases the production shield output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Factory or Manufacturing Plant in the city, the combination of the Hydro Plant and that improvement raises the production shield output to 200 percent of its base value.
- ▼ If there are both a Factory and a Manufacturing Plant in the city, the combination of the three raises the production shield output to 250 percent of its base value.
- ▼ It halves the pollution potential caused by shield production in the city. This does not affect pollution potential from other causes.

Nuclear Plant



The effects of this building are as follows:

- ▼ It cannot be built unless there is already a Factory (or the renamed equivalent) in the city. The Nuclear Plant does not, however, disappear if the Factory is later destroyed or sold.
- ▼ It cannot be built if there is a Solar Plant already in the city. Building one of those does not cause the Nuclear Plant to disappear, but does negate its effects.
- ▼ If there is a Power Plant already in the city, the Nuclear Plant negates its effects.
- ▼ If there is a Hydro Plant already in the city, the Nuclear Plant negates its effects.
- ▼ It increases the production shield output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Factory or Manufacturing Plant in the city, the combination of the Nuclear Plant and that improvement raises the production shield output to 200 percent of its base value.
- ▼ If there are both a Factory and a Manufacturing Plant in the city, the combination of the three raises the production shield output to 250 percent of its base value.
- ▼ It halves the pollution potential caused by shield production in the city. This does not affect pollution potential from other causes.
- ▼ If the city remains in Civil Disorder for more than one turn, there is a chance of a nuclear explosion each turn that the disorder continues. When the civilization discovers the advance in slot 32 (Fusion Power), this effect expires.

Stock Exchange



The effects of this improvement are as follows:

- ▼ It cannot be built unless there is already a Bank (or the renamed equivalent) in the city. The Stock Exchange does not, however, disappear if the Bank is later destroyed or sold.
- ▼ It increases the tax and luxury output of the city by 50 percent of its base value (rounded down).
- ▼ When there are both a Marketplace and a Bank in the city, the three together make the tax and luxury output 250 percent of base.
- ▼ If at some time there is only a Marketplace or a Bank—not both—in the city, the tax and luxury output becomes 200 percent of base.

Sewer System



The effects of this improvement are as follows:

- ▼ It cannot be built unless there is already an Aqueduct (or the renamed equivalent) in the city. The Sewer System does not, however, disappear if the Aqueduct is later destroyed or sold.
- ▼ A Sewer System enables the city's population to grow beyond the size specified in the Cosmic Principles. (For more details, see Step 6: The Foundations of Civilization.) The default size limit is 12.
- ▼ If the Sewer System is destroyed or sold after the city has surpassed size 12, it has no effect on the city—unless the city later shrinks below size 12, in which case a Sewer System is needed for the city to grow past 12 again.

Supermarket



A Supermarket has only one effect. It increases by 50 percent (rounded down) the Food output of every Farmland (doubly irrigated) square in the City Radius that is being worked by a citizen.



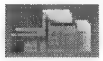
Superhighways



The effects of this improvement are as follows:

- ▼ It increases by 50 percent (rounded down) the trade output of every terrain square in the City Radius that has either a road or railroad *and* is being worked by a citizen.
- ▼ It increases the per-turn income from the city's trade routes by 50 percent. (This has no effect on trade route initiation bonuses.)

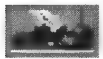
Research Lab



The effects of this building are as follows:

- ▼ It cannot be built unless there is already a University (or the renamed equivalent) in the city. The Research Lab does not, however, disappear if the University is later destroyed or sold.
- ▼ It increases the science output of the city by 50 percent of its base value (rounded down).
- ▼ When there are both a Library and a University in the city, the three together make the science output 250 percent of base.
- ▼ If at some time there is only a Library or a University—not both—in the city, the science output becomes 200 percent of base.

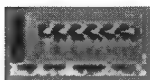
SAM Missile Battery



The effects of this improvement are as follows:

- ▼ All units in the city defend at twice their normal Defense Factor against air units (those with a Domain of 1), including missiles.
- ▼ It provides no bonus against nuclear units, even if they count as air units or missiles.

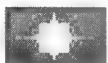
Coastal Fortress



The effects of this building are as follows:

- ▼ All units in the city defend at twice their normal Defense Factor against naval units (those with a Domain of 2).
- ▼ It cannot be built in landlocked cities—those not adjacent to at least one Ocean square.

Solar Plant



The effects of this building are as follows:

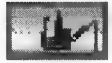
- ▼ It cannot be built unless there is already a Factory (or the renamed equivalent) in the city. The Solar Plant does not, however, disappear if the Factory is later destroyed or sold.
- ▼ The Solar Plant negates the production and pollution effects of any other energy plant (Power, Hydro, or Nuclear) in the city. Note that this does *not* remove the threat of nuclear explosion caused by a Nuclear Plant.
- ▼ It increases the production shield output of the city by 50 percent of its base value (rounded down).
- ▼ If there is a Factory or Manufacturing Plant in the city, the combination of the Solar Plant and that improvement raises the production shield output to 200 percent of its base value.
- ▼ If there are both a Factory and a Manufacturing Plant in the city, the combination of the three raises the production shield output to 250 percent of its base value.
- ▼ It completely eliminates the pollution potential caused by shield production in the city. This does not affect pollution potential from other causes.
- ▼ If there are polluted squares in your territory—within the city radii of your cities—each Solar Plant subtracts half of one of these squares from the global pollution total (which is used to determine the risk of Global Warming).

Harbor



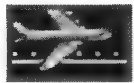
A Harbor has only one effect. Every Ocean square in the City Radius which is being worked by a citizen produces one more Food than normal.

Offshore Platform



An Offshore Platform has only one effect. Every Ocean square in the city radius which is being worked by a citizen produces one production shield in addition to whatever else it already produces.

Airport



The effects of this improvement are as follows:

- ▼ It enables units to Airlift (move in one turn) to the civilization's other cities with Airports.
- ▼ The city produces veteran air units (those with a Domain of 1, that is).
- ▼ Any air unit spending a full turn in the city returns to full combat strength (is repaired completely).
- ▼ It adds slightly to the value of any trade route between two cities that both have Airports.

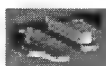
Police Station



The effects of this building are as follows:

- ▼ If any citizens in the city are unhappy due to military units being distant, this makes one of those unhappy citizens content (removes the unhappiness).
- ▼ If the civilization is under any type of government other than Democracy or Republic, the effect of this improvement is completely irrelevant.

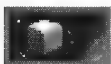
Port Facility



The effects of this improvement are as follows:

- ▼ The city produces veteran naval units (those with a Domain of 2).
- ▼ Any naval unit spending a full turn in the city returns to full combat strength (is repaired completely).

SS Structural



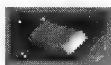
The effects of this improvement are as follows:

- ▼ It adds one Structural piece to the civilization's spaceship. The maximum number possible is 39.
- ▼ It leaves no lasting effect on the city that builds it, and therefore cannot have a non-zero upkeep cost.
- ▼ It is not available in any scenario with the Total War option toggled on.
- ▼ It is not available in any game with the Bloodlust rule activated. (This is done using the Customize Rules option during game setup; see Step 7: "Scenario" Means "Situation" for the details.)

CAUTION

Under no circumstances should you give any of the spaceship parts (SS Structural, SS Module, and SS Component) an upkeep cost. First of all, it would be pointless. Secondly, it can cause *Civilization II* to crash.

SS Component



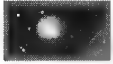
The effects of this improvement are as follows:

- ▼ It adds one Component—the player's choice of Thrust or Fuel—to the civilization's spaceship. One Fuel and one Thrust combine to make an engine. Each engine duo decreases the flight time of the ship. The maximum number possible is 8 of each.
- ▼ It leaves no lasting effect on the city that builds it, and therefore cannot have a non-zero upkeep cost.



- ▼ It is not available in any scenario with the Total War option toggled on.
- ▼ It is not available in any game with the Bloodlust rule activated. (This is done using the Customize Rules option during game setup; see Step 7: “Scenario” Means “Situation” for the details.)

SS Module



The effects of this improvement are as follows:

- ▼ It adds one Module—the player’s choice of Solar Panel, Habitation, or Life Support—to the civilization’s spaceship. One of each combine to make up a functioning population unit. Each population unit adds to the player’s final score. The maximum number possible is 4 of each.
- ▼ It leaves no lasting effect on the city that builds it, and therefore cannot have a non-zero upkeep cost.
- ▼ It is not available in any scenario with the Total War option toggled on.
- ▼ It is not available in any game with the Bloodlust rule activated. (This is done using the Customize Rules option during game setup; see Step 7: “Scenario” Means “Situation” for the details.)

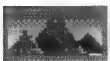
Capitalization



The effects of this improvement are as follows:

- ▼ It converts all Production in the city (shields) into Tax income.
- ▼ This improvement is an ongoing activity; it is never finished.

Pyramids



Pyramids give the Granary effect to every city in the civilization (see Granary, above, for the details).

Hanging Gardens



The effects of this wonder are as follows:

- ▼ It makes three content citizens in the city happy.
- ▼ It makes one content citizen happy in every other city in the civilization.

Colossus



The Colossus adds one trade to the output of every terrain square in the city radius which is being worked by a citizen and already produces at least one trade.

Lighthouse



The effects of this wonder are as follows:

- ▼ The effect of the unit attribute *Might be lost away from land* is ignored for this civilization; no units are lost at sea in this way.
- ▼ The movement allowance of all of this civilization's naval units (those with a Domain of 2) is increased by one.
- ▼ All new naval units built by this civilization are veteran units.

Great Library



This building grants to the civilization every advance already held by at least two other civilizations; that is, it gives the advance the moment a second civilization acquires it.

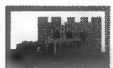
Oracle



Doubles the effect of all Temples in this civilization's cities. This increase is calculated *after* the effect of the advance in slot 56 (Mysticism) has been taken into account.



Great Wall



The effects of this wonder are as follows:

- ▼ It erects City Walls in every city in the civilization. (Note that these walls disappear when the Great Wall becomes obsolete or is captured.)
- ▼ It doubles the Attack Factor of this civilization's units versus Barbarian units.
- ▼ It forces any civilization in negotiation with this one to offer a cease-fire and a peace treaty. Note that, unlike the Eiffel Tower and most other diplomatic effects, this affects human players.

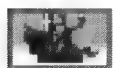
Sun Tzu's War Academy



The effects of this wonder are as follows:

- ▼ All of this civilization's cities produce veteran ground units (those with a Domain of 0).
- ▼ Any of this civilization's units that engages in and survives one combat becomes a veteran unit.

King Richard's Crusade



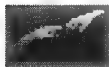
Every terrain square in the City Radius which is being worked by a citizen produces one extra production shield in addition to whatever else it already produces.

Marco Polo's Embassy



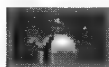
This wonder establishes an embassy with every other civilization in the game.

Michelangelo's Chapel



This wonder gives the Cathedral effects to every city in the civilization. (See Cathedral, above, for the details.) Note that this makes it unnecessary and impossible to build Cathedrals for as long as the chapel is not obsolete.

Copernicus' Observatory



This wonder increases the total science output of the city by 50 percent (rounded down). Note that this is not the same as the increases in science provided by improvements like the Library. This increase affects the *total* science output, not the *base*, meaning that all other increases are taken into account first, then multiplied by 1.5 to get the final 150 percent output.

Magellan's Expedition



This wonder increases the movement allowance of all of this civilization's naval units (those with a Domain of 2) by two.

Shakespeare's Theatre



This wonder makes all unhappy citizens in the city content.

Leonardo's Workshop



Whenever an advance makes a new unit possible, that unit takes the place of any existing units with the same Domain and Role that are made obsolete by the same advance.



J. S. Bach's Cathedral



This wonder makes two unhappy citizens content (removes the unhappiness) in every city in the civilization.

Isaac Newton's College



The effects of this wonder are as follows:

- ▼ It doubles the total science output of the city.
- ▼ Note that this is not the same as the increases in science provided by improvements like the Library. This increase affects the *total* science output, not the *base*, meaning that all other increases are taken into account first, then multiplied by 2 to get the final 200 percent output.
- ▼ If both this and Copernicus' Observatory exist in the same city (and neither is obsolete), that city's science output is effectively *tripled*.

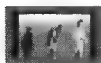
Adam Smith's Trading Co.



The effects of this wonder are as follows:

- ▼ It eliminates the upkeep cost for every city improvement with an upkeep cost of 1—in every city throughout the entire civilization.
- ▼ Improvements with any other upkeep cost are not affected at all.

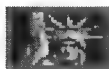
Darwin's Voyage



The effects of this wonder are as follows:

- ▼ It completes the civilization's current research project—gives the advance.
- ▼ It immediately grants the next advance.

Statue of Liberty



The effects of this wonder are as follows:

- ▼ It enables the civilization to change to any type of government, regardless of whether the prerequisite advance has been discovered.
- ▼ It eliminates the period of Anarchy between governments. The change in government takes place immediately.

Eiffel Tower



The effects of this wonder are as follows:

- ▼ It gives a one-time improvement to the civilization's reputation rating.
- ▼ It accelerates the gradual improvement over time of every other empire's attitude toward the civilization.

Women's Suffrage



This wonder gives the Police Station effects to every city in the civilization (see Police Station, above, for the details). Note that this makes it unnecessary and impossible to build Police Stations for as long as suffrage is not obsolete.



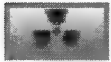
Hoover Dam



The effects of this wonder are as follows:

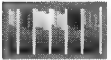
- ▼ It gives the Hydro Plant effects to every city in the civilization (see Hydro Plant, above, for the details). Note that this makes it unnecessary and impossible to build Hydro Plants for as long as the dam is not obsolete.
- ▼ Note that the Hoover Dam has no effect on any city that has a Solar Plant. The Solar Plant supersedes the effects of the Hoover Dam, just as it would negate the effects of a Hydro Plant.

Manhattan Project



This wonder enables all civilizations that have discovered the necessary advances to begin building nuclear units. No civilization can build nuclear units until this wonder has been completed.

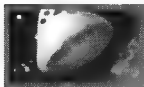
United Nations



The effects of this wonder are as follows:

- ▼ This civilization establishes an embassy with every other civilization in the game.
- ▼ Under a Democracy, there is a 50 percent chance that the leader of this civilization will overrule the Senate and start a war.
- ▼ It forces any civilization in negotiation with this one to offer a cease-fire and a peace treaty. Note that, unlike the Eiffel Tower and most other diplomatic effects, this affects human players.

Apollo Program



The effects of this wonder are as follows:

- ▼ It enables all civilizations that have discovered the necessary advances to begin building a spaceship.
- ▼ It reveals the entire map to every civilization.

SETI Program



This wonder gives the Research Lab effects to every city in the civilization (see Research Lab, above, for the details). Note that this makes it unnecessary and impossible to build Research Labs for as long as the program is not obsolete.

Cure for Cancer



In every city in the civilization, this wonder makes 1 content citizen happy. ∞

STEP 6

THE FOUNDATIONS OF CIVILIZATION

THIS CHAPTER IS FULL OF THINGS that even the designers at MicroProse hesitated to mess with. In some cases, that was because of the very real possibility of severely unbalancing the scenario. In others, it was just that it didn't occur to us.

You need to have no such compunctions. I explain how to safely modify the powerful “cosmic principles” you'll find at the beginning of *rules.txt*, the text *Civilopedia*, and all of the menus, pop-up boxes, and other text that appears in the game. These are some of the little touches that make a scenario complete.

Cosmic Principles

There's a collection of simple numbers that have far-reaching effects on *Civilization II*—and on your scenario. In *rules.txt*, they're referred to as the “cosmic principles.” That's as good a name as any. They're the universal constants.

You really want to be careful with these. It's not that you run a great risk of crashing the game when you modify the cosmic principles; you certainly can, but that's not the point. Ill-considered changes will unbalance your scenario pretty severely. Think before you act.

NOTE

I say, “be careful,” maybe a million times in this book, and you've probably stopped listening by now, but it's true every time. If you go around changing things recklessly, you're almost guaranteed to end up with a really awful scenario that no one has any fun playing. Never mind that you can make the game crash a lot, too.

This section presents the cosmic principles in some detail. The better you understand them, the less likely you are to seriously botch up the balance of your scenarios when you change them.

FANTASTIC WORLDS

If you have *Fantastic Worlds*, you can modify the cosmic principles using the Effects Editor on the Editors menu. This method is likely to be a little easier for most folks.

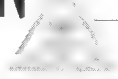
The cosmic principles are defined in *rules.txt*. You don't have to search to find this line; it's right at the beginning:

```
@COSMIC
```

That's the header marking the beginning of the section. Each of the lines of text following it—down to the line that determines the *Max Effective Science Rate in Fundamentalism*—sets one of the cosmic principles. Only the number before the semicolon (and the semicolon itself) really count. Everything after the semicolon is what programmers call “comment” text. It's there to tell us what's what, but it's completely ignored by the *Civ II* program.

All the lines that start with the @ symbol are header lines. Do not ever change any of the header lines. If you do, *Civilization II* will probably crash. Also, you should never add lines to or delete lines from *rules.txt*—not even empty lines. In some cases, the program counts lines to find data, and if you remove or add something, you can only cause trouble.

CAUTION



The cosmic principles are rather important, so let's go through them one at a time.

Road Movement Multiplier



This number determines how many squares a ground unit can move along a road or river by spending one movement point. So, for example, the default value of 3 means that units moving from one road or river square to another use only one-third of a movement point doing so. Setting this number to 1 or to 0 (zero) has the same effect—roads have no movement effect in both cases.

Note that this also changes the number of squares that units with the attribute *Treats all terrain as road* can move in a turn. Whatever this number, that's the number of squares those units are able to move each turn.

I've never known any designer to increase this one and not bring it back down later, after some testing. It seems that lessening the overall movement of units is always safer, in game balance terms, than increasing it—unless your scenario is so large and spread out that fast units don't threaten the balance of power. Slow movement is already an accepted part of *Civilization II*, and the ability to move too far too fast should be reserved for later in the game. There are ways to make units fast that you have more focused control over.

Chance of Losing Ships



Whatever naval units you assign the *Might be lost away from land* attribute, this number determines the chance that they'll be lost. Specifically, if you think of this number as x , there is a one in x chance ($1/x$) of any individual unit being destroyed. So, increasing this number makes it *less* likely that such units disappear. The only

way to make the loss of units *more* likely is to lower this number to 1, guaranteeing disappearance.

You should never set this number to zero. Trying to get a computer to divide by zero is always a bad idea. (In fact, it's such a bad idea that there's an error message named for it.)

CAUTION



Remember that unless you remove the advances in slots 75 (Seafaring) and 57 (Navigation) from your scenario, they will have their usual mitigating effects on this threat. The Lighthouse wonder (or whatever you rename it) eliminates the problem for the single civilization that builds it.

Players really hate losing units to random chance (or a mistaken keystroke). If units get lost every time in your scenario (1/1), you'll force the players to be *very* careful with their sea units. Lessening the chance, on the other hand, allows greater latitude in sea exploration at times when there are only unreliable naval units. If, just to take one example, you decided that *all* sea units are inherently undependable, it would be wise to balance the increased number of sinkable ships with a lower chance of losing each one. However, if your intention is to prohibit ocean travel completely, you can do so, but I'd suggest adding some text to the scenario explaining the background for this situation. (The sea monsters out there are breeding like flies, maybe.) How to do so is covered in the section "Game Text" later in this step.

Citizens Eat

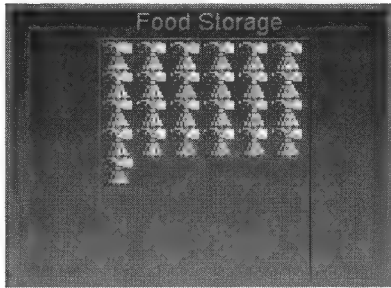


Technically, this determines how many food units each "citizen"—really each citizen *icon*—in a city uses as maintenance each turn. What it really controls, however, is the rate of city growth. If you lower this number, there will be more surplus food and thus faster population growth in *every*

city. By raising this number, you can stunt or even stop the growth of cities. It's a bad idea to make this number zero.

Unless you mitigate the change by modifying other food factors—how much food the terrain types produce, the number of rows in the food box, and so on—messing with this can really unbalance a scenario. Cities will grow either too quickly or barely at all, and both of these situations cause problems. Providing too much food is, however, a good way to free up population for other tasks—if you've made money, science, or happiness more difficult to attain in your scenario. Just remember—test, test, test. All the best designers know that if you don't play your game yourself, chances are that no one else will want to, either.

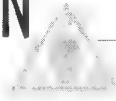
Rows in Food Box



This constant controls the size of the Food Storage box in every city—how much food must accumulate for the city to grow. The box contains one column for every point of population and exactly the number of rows you specify here. Thus, the size of the Food Storage box is this number multiplied by the size of the city.

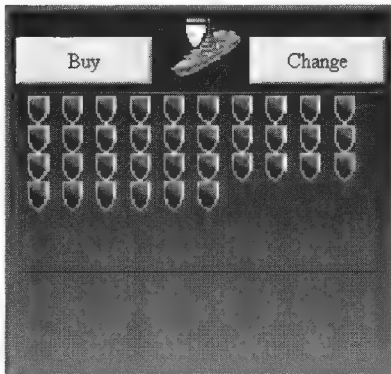
CAUTION

Do not set this constant to zero.



This is another one of the factors controlling the rate of city growth. The higher you make this number, the slower cities grow. Unless these food factors are balanced, the scenario probably isn't. Cities will either grow too quickly or not at all, and both of these are problems.

Columns in Shield Box



This cosmic principle controls the size of the Production box in every city; it's the number by which the cost of every unit and improvement is multiplied to determine how many shields must accumulate to finish the item. The text associated with this one in *rules.txt* is misleading. This constant doesn't determine the number of rows in the box; rather, it determines how many shields are in each row—the number of *columns* in the box.

CAUTION

Do not set this constant to zero.



CAUTION

The cost of a unit or improvement—the number that this constant is multiplied by to get the true cost in shields—is also under your control. Refer to Step 3: Design Units and Step 5: Make Improvements for the scoop.

This is quite useful when you want the time scale of your scenario to be more realistic than the *Civ II* timeline. For example, each turn in the early game is 50 years long. Only the fact that it's a game, not reality, allows it to make sense that it takes hundreds of years to “build” a group of Warriors or a Phalanx. If your scenario progresses month by month or one year at a time, you can manipulate this number (and other production factors, like the output of terrain types) to make the production schedule as realistic as you want it.

Settlers Eat

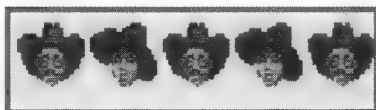


The next two numbers control how much food every Settlers-type unit (Role = 5) takes as maintenance each turn from its Home City. The first is relevant only to cities under the early forms of government—Despotism, Monarchy, and Anarchy. The second affects cities under Communism, Fundamentalism, Republic, and Democracy. You *can* set either or both of these to zero (0).

Changing Settler maintenance has an effect on the growth of cities, but not nearly as significant an effect as some of the other food-related constants. If you raise/lower this number, there will be less/more surplus food and thus slower/faster population growth, but you leave the player much more control over the effect than with the other constants. Building units is a choice, and if the Settler-type units don't hang around long, the upkeep isn't paid for enough turns to become onerous. Also, the Settler units are disbanded before any population is lost from cities.

If you want to make city growth tougher or easier to achieve, this is probably the fundamental constant to change that will have the least unbalancing effect on your scenario.

Unhappiness Factors



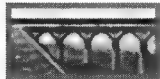
After the two Settlers Eat lines come the two lines that affect citizen unhappiness. These both enable you to make cities in the game more or less difficult to manage, especially if you've limited the types of government available or prevented government switching altogether.

The first, City Size for First Unhappiness at Chieftain Level, controls how large a city can get before citizens join the roster unhappy. If you remember the manual, it tells you that at Chieftain difficulty, the first 6 citizens are content, but every one after that starts out unhappy and must be mollified somehow. This number is what makes it that way. The default for this number is 7, and the seventh citizen is unhappy. At each higher difficulty level, this number is reduced by one. It's a bad idea to make this number lower than 5; otherwise, in a Deity-level game, it would be less than zero.

The second of these, Riot Factor Based on # Cities, is similar. When there are more than a certain number of cities in an empire, the sheer size of the nation starts to make citizens uncomfortable, and therefore unhappy. This effect is less noticeable than the other one, but no less of a factor in a long or a large game. The number you put here is the number of cities at which the extra unhappiness begins. So, using the default as an example, an empire with 13 cities is fine, but one with 14 has trouble. Setting this to zero is not a good idea.

What these two really do is make the game more or less management intensive and—for military-minded players—frustrating. When greater resources must go into keeping the populace content, the pace of the game slows. The opposite situation lets the military and space-directed efforts roll right along. The key for scenario builders is that there's not often a reason to change one of these and balance it by changing the other in the opposite direction. (There are always exceptions.) You'll normally find yourself increasing or decreasing both at once, so be careful that you don't cause too great an effect.

Size Limits



No city in your scenario can expand beyond the size you set here, unless that city builds a particular improvement. There are two of these, one each for the improvements in slot 9 (Aqueduct) and slot 23 (Sewer System). You should never set either of these numbers to zero, and setting either of them lower than 3 or 4 makes it almost impossible for cities to grow large enough to build the improvements that allow them to grow larger. That's a problem.

Now, a city's population is not *decreased* if these improvements aren't in place. Once a city grows past the size limit, the improvement has had its effect and that's that. You can set a city's beginning size above the limit, and that city will have no need for these improvements unless it somehow shrinks below one of the limits. So, for example, you could limit the size of the player's cities, but not those of the AI civilizations—or vice versa.

It's not normally useful to make the first of these numbers higher than the second. No one can build the second improvement in any city until the first one already exists there. Therefore, a city would need to build both to grow beyond the size limit set by the second improvement. A city would then always overcome both limits at the same time (as soon as the second improvement is finished).

Tech Paradigm

This number sets the default pace of research. The higher you set this constant, the more science (beakers) each advance costs, and the slower research progresses. Lower numbers make science go more quickly. Do not set this to zero.

You can change this number if you like, but there's little point. The Scenario Parameters option on the Cheat menu includes a feature that enables you to set the Tech Paradigm for a scenario. That setting supersedes this one.

Transform Time



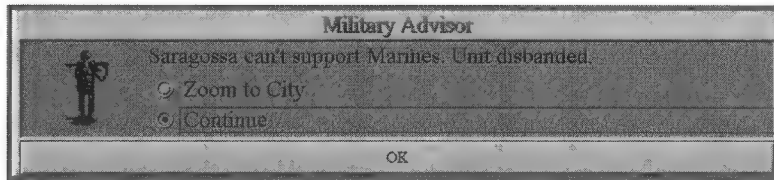
This constant simply determines half of how many turns of work it takes a single Engineer (any unit in slot 2 with a Role of 5) to transform one type of terrain to another. Multiply this number by 2 to get the actual number of turns. Do not set this constant to zero.

NOTE

The transformation possibilities are set in the terrain entries of *rules.txt*. For the detailed scoop, refer to Step 2: Create Your Map.

What's the point in messing with this one? For most scenarios, especially historical ones, there is none. However, if you want to allow Engineers to turn land into ocean (Panama Canal scenario, anyone?) or terraform alien worlds (Alpha Centauri?), how long it takes them could turn out to be a vital part of the timing in your scenario.

Support



There are three lines in a row that determine—for specific forms of government—the number of units every city can support without paying for it (in shields). Every unit above this number that a city supports costs one shield per turn. You can set any of these to zero.

This gives you some control over the extent of military might available under Monarchies, Communisms, and Fundamentalisms. (Despotisms and Anarchies pay unit support based on city size; Republics and Democracies pay support for every unit.) Let's face it, though; a dedicated warlord player will always find a way around support issues. This just enables you to make it more or less expensive to wage wars.

Balancing these numbers is one of the best ways to mitigate the military advantage of Fundamentalism.

Communism: Palace Distance

This is the reason cities under Communism suffer no corruption or waste (in the game, that is). Both of these afflictions are computed, for every government type, based on a city's distance from the capital of the civilization (the city with the Palace in it). All cities in a Communist empire are treated as being exactly the same distance from the capital, and this number determines what that distance is. A higher number here makes for more corruption and waste. A lower number, like the zero default, lessens these problems.

If you choose to make Communism more realistic by adding corruption and waste, that's fine. Just make sure you test the scenario. If it's unbalanced by this change, you might need to give Communism a little boost in some other category (unit support, for instance).

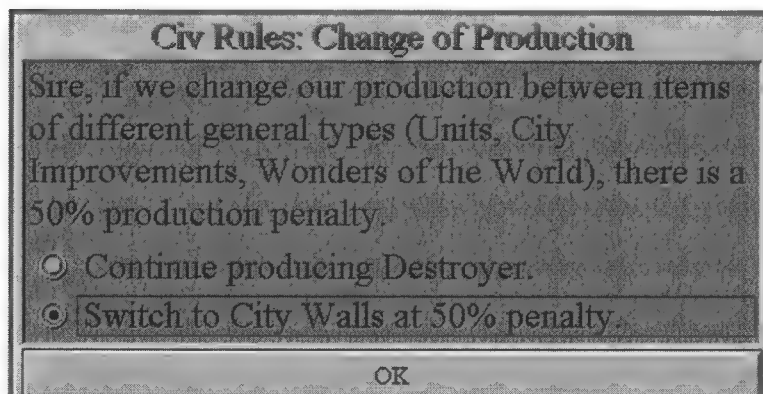
Fundamentalism: Lost Science

No matter how much science a Fundamentalist empire generates, a percentage of it simply disappears off the top—lost to inquisitors or fear of heresy or plain intolerance. This number determines that percentage. (If you want to, you *can* set this number to zero.)

This is supposed to be the balancing factor that makes up for the income and military support advantages of the fundamentalist form of government. It doesn't work that well. Fact is, one of the most common strategies for defeating *Civilization II* involves Fundamentalism. If you intend to leave this form of government in your scenario, you should seriously consider raising this percentage to balance things.

However, if you take away the military support advantage (see the Support cosmic principle to learn how), you might end up lowering this percentage so that Fundamentalism doesn't become so useless that no one considers using it.

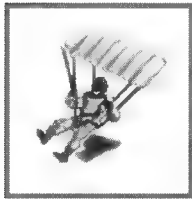
Penalty for Changing Production



One of the ideas introduced in the changeover from *Civilization* to *Civilization II* was that of losing production when you switch from one type of project to another. This constant controls what percentage of the accumulated shields disappear. It's relevant to every difficulty level except Chieftain. If you want your scenario to reflect the state of affairs in the original *Civilization*, you can set this to zero.

It's unlikely that you'll run into a situation in which it's necessary to your scenario to have this set to a specific value. If you want to change this, feel free. The risk of unbalancing your scenario with this one is pretty low.

Paradrop Range



The jump range (in terrain squares) of units with the *Can be ordered to paradrop* attribute is determined by this number. If you set this to zero, paradropping becomes totally useless. (A better way to achieve this effect, however, is to take away the attribute from all the units in the scenario.)

This is one that I've modified myself, in my Mars scenario. In that case, the idea was that certain units could drop from orbit, so I made the range about half as big as the world was wide. That way, the units could drop almost anywhere. In most cases, you won't want to give players that much latitude. Paradropping is another one of those features that players can use to get an advantage over the AI civilizations. The range limit is meant to simulate the flight radius of the jump plane and the dangers of flying into enemy territory.

If you're working on a scenario in which realism isn't an issue, paradropping can become any type of instant transportation you can imagine—the more appropriate to the milieu it is, the better.

Mass/Thrust Paradigm

This constant determines (to a certain extent) how quickly spaceships travel. The higher this number is, the slower the ships get to their destination. Don't set this to zero.

There's one big, fat astronomical mistake in *Civilization II* (and the original *Civilization*, for that matter) that I can't resist pointing out. Alpha Centauri is one star of a *triple* star system.

The chance of there being *any* planets there, much less ones humans can colonize, is about zero. If you had a choice, you'd send your colony ship somewhere else.

NOTE

When you're designing a scenario, this is rarely something you need to fuss with. I'm sure you can think of situations in which it would be important for spaceships to move more or less quickly, but they're pretty specific. If the thought leads into an interesting scenario—build it.

Fundamentalism: Max Science

This number determines how high a Fundamentalist civilization can set the science portion of its Tax Rate. The rate is measured in increments of 10 percent, and so is

this constant. Thus, for example, if you changed this to 3, a Fundamentalism would not be able to funnel more than 30 percent of its trade income to research. It's okay to set this one to zero.

CIV II

If you have only the original version of the game and none of the patches MicroProse has released or either of the scenario packs, then this last cosmic principle might not appear in your version of the *rules.txt* file.

Considering that you can also control what percentage of a Fundamentalism's science is lost (presumably to the enforced ignorance that prevails under religious governments), this might seem like overkill. However, since *Civilization II* was released, plenty of players have found ways to get around the science limitations of the Fundamentalist government. Combined with the military advantages, this makes world conquest almost easy.

The other side of this constant is that it allows you to *raise* the amount of research a Fundamentalism can do. Why would you want to do that? Well, it's one way to give one of your AI civilizations an advantage. (Of course, in that situation, you'd want to prevent human players—the other civilizations—from using Fundamentalism.) Also, you might have reduced the military advantages of this form of government, and raising this is one way to balance that.

The Text Civilopedia

If you're serious about your scenario, you can't avoid editing the Civilopedia. Especially when you've made extensive changes, you take the chance of losing your players when they can't figure anything out. The Civilopedia is the player's reference of first resort, because it's available in-game, where it's too convenient to ignore.

Now, I don't mean to imply that you can edit the *real* Civilopedia. As those of you who have read the manual for either of the scenario packs know, in a scenario, the usual Civilopedia never functions. What we're discussing here is the alternate—the text only Civilopedia.

Editing Guidelines

To edit the text *Civilopedia*, open the file *pedia.txt* (see Figure 6-1). Before you start running rampant around the file, read these few important rules:

- ▼ Never touch a line that begins with the @ symbol and has text on it that's ALL CAPITALIZED. Just like in every other *Civ II* file, these are headers and programming-type commands. You can only do harm by messing with them.
- ▼ There are lines in *pedia.txt* that begin with @ and can be edited. Every one of them includes an equals sign (=), and you can only edit the text *following* the equals sign.
- ▼ Any line that begins with a semicolon (;) is what programmers call a “comment” line. The program ignores these completely. Thus, you can put any text you want on these lines, as long as you do not remove the semicolon.
- ▼ Part of a line can be a comment, too. If there is a semicolon somewhere in the text on a line, all the text on that line after the semicolon is considered “comment” and ignored by the program.
- ▼ Do not remove or add lines—not even empty lines. You can remove comment lines if you really want to, but why bother?
- ▼ Any word that begins with the percent sign (%) is *not text* and you definitely shouldn't edit it. (Generally, each of these is a call to insert the value of a variable.)

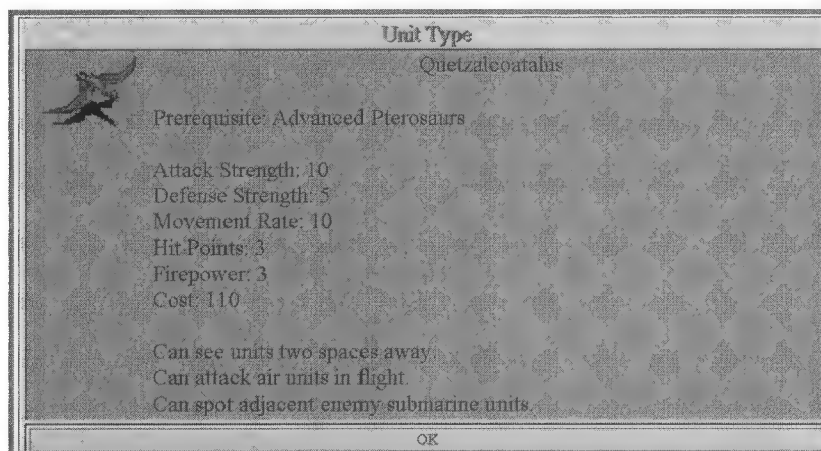


Figure 6-1:
A sample entry from
the text *Civilopedia*

What You Can Do

A good portion of the text Civilopedia is drawn directly from *rules.txt*. That's pretty convenient for scenario designers, because it means that much less text we need to rewrite. However, it also limits the amount of explanation we can offer—we can't always present the player with the whys and wherefores. This is especially true of units and advances; the entries for them are drawn almost completely from *rules.txt*.

NOTE

There are no entries for Terrain Types or Game Concepts in the text Civilopedia.

What can you edit? The entries in the *pedia.txt* file include (in order):

- ▼ Civilopedia window titles
- ▼ Some text relevant to advances
- ▼ Descriptions of unit attributes
- ▼ Entries for City Improvements and Wonders of the World.
- ▼ Descriptions of governments

As you edit, you have a few formatting tools available. It's not much, but if you're creative and clever, you can do a lot with a little.

- ▼ A line with nothing on it but a caret (^) is displayed as a blank line.
- ▼ A line with nothing on it at all is not displayed.
- ▼ A line of text that begins with two carets (like this: ^^) is centered in the on-screen box.
- ▼ No special characters are available—just plain text. (That's why bulleted lists are made with asterisks—best we could do.)
- ▼ If a text box is the wrong size, you can edit the number on the associated `@width` line to widen or narrow the box.
- ▼ If you need to put the percent symbol (%) in the text for a City Improvement or a Wonder of the World, you must put two instead (%%).

Section by Section

The identity of each piece of data in *pedia.txt* is not immediately obvious. Table 6-1 lists the file header by header and tells you what’s what.

Table 6-1. Text Civilopedia Sections

HEADER	DESCRIPTION
@PEDIAPICKCIV	This controls the box that appears when you select Civilization Advances from the Civilopedia menu—the listing from which you choose an advance entry to view. You can change the width of the box and the title displayed at the top. You can also change the number of columns in the box, but there’s no real reason to.
@PEDIAACIV	Here you can modify the width and title of the box in which the listing for every advance is displayed.
@PEDIAACIVFACTS	Each of these lines of text appears in the entry for one or more advances. In most cases, you cannot change the actual effects described or which advances cause each effect; both the effects and the text are linked to specific advance slots. You can, however, edit the text itself. Note that the line “Cancels the effect of” is used to mark the advances that make wonders obsolete, and the name of the wonder is appended to this text.
@PEDIAPICKUNIT	This controls the box that appears when you select Military Units from the Civilopedia menu—the listing from which you choose a unit entry to view. You can change the width of the box and the title displayed at the top. You can also change the number of columns in the box, but there’s no real reason to.
@PEDIAUNIT	Here you can modify the width and title of the box in which the listing for every unit is displayed.
@PEDIAUNITFACTS	Each of these lines of text describes one of the unit attributes. The text is included in the entry for every unit that has that attribute.
@PEDIAPICKGOVT	This controls the box that appears when you select Governments from the Civilopedia menu—the listing from which you choose a government entry to view. You can change the width of the box and the title displayed at the top.

<code>@PEDIAPICKIMPROVE</code>	This controls the box that appears when you select City Improvements or Wonders of the World from the Civilopedia menu—the listing from which you choose an improvement or wonder entry to view. You can change the width of the box and the title displayed at the top. You can also change the number of columns in the box, but there's no real reason to.
<code>@PEDIAIMPROVE</code>	Here you can modify the width and title of the box in which the listing for every improvement and wonder is displayed.
<code>@PEDIAIMPROVE#</code> (1 to 66)	Each of these sixty-six numbered entries describes one of the City Improvements or Wonders of the World. (Note that they're numbered according to slot.) You can change only the text; the width is set in <code>@PEDIAIMPROVE</code> , and the title is always the name of the improvement, taken from <i>rules.txt</i> . (Notice that the line immediately preceding the header is only a comment; it's a good place to note the title.) You can't change the effects of each improvement, so unless you intend to mislead the player, it's a good idea to leave the descriptions' meaning intact.
<code>@PEDIAGOV</code>	This text is the entry for Governments: Overview. You can change the width of the box, the title, and the text itself.
<code>@PEDIAGOV#</code> (0 to 6)	Each of these seven numbered entries (<code>@PEDIAGOV0</code> through <code>@PEDIAGOV6</code>) describes one of the types of government available in the game. For each, you can change the width of the box, the title, and the text. You can't change the effects of each government much, so it's a good idea to leave the descriptions mostly intact—unless you intend to mislead the player.

Menus

What the manuals don't tell you—and neither do any of the other books about the game—is that you can change all of the text on all of the menus in *Civilization II*. Is this helpful when building a scenario? Well, kind of. It's fun, anyway.

I'll let one example spark your imagination, then get right into the logistics. Let's say you're working on a scenario in which there are no cities—like in the X-COM Assault scenario, you've turned them into bases of some sort. Wouldn't it be nice if the menus reflected this change? The City Report Options and City Status features could be Base Report Options and Base Status, instead. It's the thoughtful, little touches that help to establish and maintain the atmosphere of a scenario.

Fantastic Worlds adds a section to this file for the Editors menu. Yes, you can edit that section, but my point is that those of you who have this scenario pack are starting out with a different version of this file. If you include an edited version of this file in your scenario folder, that scenario is *not* compatible with earlier versions. You'll need to modify it for use on any computer that doesn't have *Fantastic Worlds*.



To edit the menus, open the file *menu.txt*. You can really cause trouble mucking around in this file, so before you do anything, read these important rules. Some you've heard before—applied to the text Civilopedia and other text files—and some are new.

- ▼ Never change a line that begins with the at symbol (@). Just like in every other *Civ II* file, these are headers. You can only do harm by messing with them.
- ▼ Any line that begins with a semicolon (;) is what programmers call a “comment” line. The program ignores these completely. Thus, you can put any text you want to on these lines, as long as you do not remove the semicolon.
- ▼ Do not remove or add lines—not even empty lines. You can remove comment lines if you really want to, but why bother?

- ▼ You'll see an ampersand (&) on most lines. That's a marker telling the program to underline the very next character. The underline is a way of letting players know the keyboard shortcut for that menu or option. You can't change the shortcuts, so it's wise to make sure the same letter is underlined, regardless of what you do to the text. If you somehow remove that letter altogether, it's best to remove the underline, too.
- ▼ Many lines include a vertical line like this: | (it's called the "pipe" character). That's a signal to the program that everything on the line after the pipe goes on the right-hand side of the menu. Again, this is a way to communicate keyboard shortcuts to players. You can edit the text after the pipe, but it doesn't change the keyboard shortcut. It might confuse the player, though.
- ▼ There's one last thing you should know. Editing the Orders menu portion of this file doesn't always have the effect you expect. That's because another file interferes. The text in *labels.txt* (which I cover in the very next section) supersedes the text of *menu.txt* in a few cases, all of which happen to be on the Orders menu.

Game Text

Another unadvertised fact is that you can change quite a lot of the incidental text in *Civilization II*. It's probably not the first thing you'll want to do (even the designers at MicroProse didn't start messing with this text until we were working on the scenarios for *Fantastic Worlds*), but once your scenario starts to take shape, you'll have some very definite ideas of what needs changing. What's in these files? We're talking about every pop-up message and option box in the game, all the negotiations dialogue, and more.

No more preamble; let's get into it.

Editing Guidelines

When you're changing game text, there are two files you can work with: *labels.txt* and *game.txt*. I'll get into each separately a little further on, but there are (as usual) a few cautions that you really need to know about.

Fantastic Worlds installs new versions of both these files. If you edit either of these for one of your scenarios, that scenario will not be compatible with earlier versions of *Civilization II*. You'll need to modify the file for use on any computer that doesn't have *Fantastic Worlds*.

FANTASTIC
WORLDS

Before you start running rampant around the files, read these few important rules. These are quite similar to the rules for other text files, and it might seem redundant to repeat them over and over, but the minor differences are important. Each of these files adheres to its own programming conventions, despite the similarities.

- ▼ Never touch a line that begins with the at symbol (@) and has text on it that's ALL CAPITALIZED. Just like in every other *Civ II* file, these are headers and programming-type commands. You can only do harm by messing with them.
- ▼ There are lines in *game.txt* that begin with @ and can be edited. Every one of them includes an equals sign (=), and you can only edit the text *after* the equals.
- ▼ Any line that begins with a semicolon (;) is what programmers call a “comment” line. The program ignores these completely. Thus, you can put any text you want to on these lines, as long as you do not remove the semicolon.
- ▼ Part of a line can be a comment, too. If there is a semicolon somewhere in the text on a line, all the text on that line after the semicolon is considered “comment” and ignored by the program.
- ▼ Do not remove or add lines—not even empty lines. You can remove comment lines if you really want to, but why bother?
- ▼ Any word that begins with the percent sign (%) is *not text* and you definitely shouldn't edit it. (Generally, each of these is a call to insert the value of a variable.)

You can't change the circumstances that bring a piece of text onto the screen, so make sure that the text you put in each spot is relevant.

Labels.txt

The *labels.txt* file is the lost and found of *Civilization II*. Basically, any text that isn't anywhere else is probably in here. There's no reason you should be able to make sense of the jumble of text in this file; it's not really organized. If you've played *Civilization II* enough, you'll be able to recognize quite a lot of the messages and parts of messages. What you don't recognize, you probably shouldn't change.

I'm not going to go through the file section by section like I do for some of the other text files. For one thing, it would make for some seriously tedious reading—thirty or so pages of line-by-line listings. For another, more than half the fun I had editing this file was in the process of discovery. Looking through this file sparked more ideas than I knew what to do with—the adjectives for all the rulers' attitudes are in here, as well as the descriptions of your reputation. I hope it inspires you, too.

FANTASTIC WORLDS

Fantastic Worlds installs a new version of this file. If you edit this for one of your scenarios, that scenario will not be compatible with earlier versions of *Civilization II*. You'll need to modify the file for use on any computer that doesn't have *Fantastic Worlds*.

One tactic that worked well for me is this: when you get a few minutes and you feel like working with *labels.txt*, just open the file and start skimming through it. When you see something that you want to change for your scenario, go right ahead and change it. When you stop skimming, write down the last line you read. That way, you can come back later and pick up where you left off. Eventually, you'll get through the whole thing.

Game.txt

The *game.txt* file is probably the most important text file in *Civilization II*. It contains most of the “popup” boxes (those boxes that pop up in the middle of the screen), menus, and a lot of the other miscellaneous text that you’ll see during a game. It’s a heck of a lot more organized than *labels.txt*, but it’s also several times larger.

In fact, *game.txt* is so large (over 4,000 lines of text) and contains so many different things that I can’t possibly point them all out or explain them all in this chapter. (My editor would have a fit.) After you’ve played *Civilization II* for a few hours, you’ll recognize most of the bits of text for what they are. Put on your wading boots and start right in. I’ll let one example from my own experience suffice to get you going.

In the Age of Reptiles scenario, I replaced all of the usual *Civ II* units with dinosaurs. When I went back and played my scenario (all good designers play their own), I noticed that the messages popping up didn’t match the setting. “Your aircraft has run out of fuel,” just didn’t work for a *pterosaur* that had exceeded its range. A little research led me to the `AFUEL` header in *game.txt*. I changed the text for aircraft crashing, but as I did, I happened to read a few of the messages around that one. Pretty soon, I was hip deep in the file, changing all the explanations of why units did and didn’t do certain things. After that, I made a habit of browsing through the file when I got free time. I discovered the diplomatic messages one morning, and I spent the rest of that day fiddling with them. This file is just plain fun.

Fantastic Worlds installs a new version of this file. If you edit this for one of your scenarios, that scenario will not be compatible with earlier versions of *Civilization II*. You’ll need to modify the file for use on any computer that doesn’t have *Fantastic Worlds*.



If you follow the editing guidelines I listed earlier in this section, you shouldn't cause any problems. As you edit, you have a few formatting tools available. It's not much, but if you're creative and clever, you can do a lot with a little.

- ▼ A line with nothing on it but a caret (^) is displayed as a blank line.
- ▼ A line with nothing on it at all is not displayed.
- ▼ A line of text that begins with two carets (like this: ^^) is centered in the on-screen box.
- ▼ No special characters are available—just plain text.
- ▼ If a text box is the wrong size, you can edit the number on the associated `@width` line to widen or narrow the box.
- ▼ You can change the title of on-screen boxes by changing the text on the associated `@title` line.
- ▼ If you need to put the percent symbol (%) in the text, you must put two instead (%%).

Have fun. ✂

STEP 7

"SCENARIO" MEANS "SITUATION"

OKAY, NOW YOU'VE DONE ALL THE PREPARATION and designing your scenario needs; the groundwork is complete. You've gone through all the files and meticulously built every tiniest piece of the puzzle. Now it's time to start putting the pieces together into a unified whole. It's time to actually build the scenario.

A scenario is not just a map and a collection of new units, reshuffled advances, renamed city improvements, pictures, sounds, and rules. A scenario is an interesting situation that's (hopefully) fun to play. The new things you've designed and created are important, but what you do with them is just as important, if not more so. As I've said before, I can't tell you how to build your particular scenario. What I can do, and what this chapter does, is lay out the tools you have and the rules to follow and give you some advice garnered from experience. Let's get to it.

Random Start Scenarios

If you have *Fantastic Worlds*, there's an exciting possibility you can build into your scenarios. You can set up multiple different situations—different scenarios based on the same group of files—and have the game choose one at random. That is, when a player decides to load the scenario, he or she ends up playing one of the alternates you have created, and which one is determined randomly. For a detailed description of how it should be done, read the relevant section of Chapter 10: Tricks of the Trade.

Defining the Tribes

Before you create the game that will become your scenario, you should first define the tribes—the potential civilizations—that you intend to have running loose in your world. You can change the names and attitudes and just about everything else about a civilization using the Cheat menu, so why is it so imperative that you do this before creating the game? It's not. There's nothing in here that you can't change after you've started the game. However, if you already know what you have in mind, getting it in ahead of time is just better organized.

So how do you do this? As with most other things, it's all in *rules.txt*. Near the end of the file is this line:

@LEADERS

That's the header marking the beginning of the section containing the definitions of all the tribes. Each of the lines of text following it—down to and including the line that begins with **Sitting Bull**—determines the characteristics of a single tribe. (The Arabs and Incas are not used.)



CAUTION

You know by now not to mess with the lines that start with the @ symbol, because they're header lines. You should also know by now not to add lines to or delete lines from *rules.txt*—so don't.

Just as I did for all the other types of entries, I'm going to take one of these definition lines apart. The Romans are defined as:

```
Caesar,Livia,0,1,1,1,Romans,Roman,0,1,1,1,Dictator,Dictator,2,Imperator,Imperatrix
```

Each part of the line is a data field, and each field except the last one must end with a comma. Table 7-1 covers the meanings of all the fields.

You can edit most of the tribe fields using the Tribe Editor in *Fantastic Worlds*. The rest, you can get to through the Cheat menu.

Table 7-1. Tribe Data Fields for Roman Example

VALUE	FIELD NAME	EFFECT
Caesar	Male Leader	This is the name of the default male leader of the civilization. If you set the Gender toggle for this Tribe to zero (0), this is the name of the leader whenever the AI controls the civilization from the start of the game. Unlike most of the names in <i>Civilization II</i> , this can be up to 20 characters in length, but you still should use only letters, numbers, and spaces in the name.
Livia	Female Leader	This is the name of the default female leader of the civilization. If you set the Gender toggle for this Tribe to one (1), this is the name of the leader whenever the AI controls the civilization from the start of the game. Unlike most of the names in <i>Civilization II</i> , this can be up to 20 characters in length, but you still should use only letters, numbers, and spaces in the name.
0	Gender Toggle	This determines whether the leader of the tribe is male or female—which controls the default name and titles. The player can override this setting for one civilization (the one the player chooses to rule) when the game begins. The only allowed values are zero (0) and one (1); do not put any other value in this field.
1	Color	Each civilization has a color associated with it, and this field determines that color. If you look in the file, you'll notice that the tribes are listed in order by color, 1 through 7, repeated three times. This order is important to the correct functioning of the game; you should never change or rearrange these numbers. Rather, if you want a particular color associated with a specific tribe, define the tribe on the line where that color already is. Also note that only one tribe of any color can be active in the game at a time. The colors are listed in Table 7-2.

Table 7-1. Tribe Data Fields for Roman Example (continued)

1 1	City Style	This number determines the set of city icons—the city style—used to show the cities of this tribe when the AI controls it. There are four possible styles (0-3), numbered to correspond to the top four rows of icons in the <i>cities.gif</i> file (For more details see the section Creating Cities later in this chapter). Note that all post-industrial and modern cities look the same.
Romans 1	Tribe	This is the name of the civilization. As with the names of most things, this shouldn't be more than 15 characters, and you should use only letters, numbers, and spaces in names.
Roman 1	Adjective	This is normally the adjectival form of the name of the civilization. However, it doesn't have to be. In the game, this text is generally applied to things a civilization owns—"Roman Scientists" is one example. Whatever text you think is appropriate will do—"Arrogant Scientists," for example.
0 1	Attack	This and the next two fields control the "personality" of the tribe when it is controlled by the AI. <i>Attack</i> determines how aggressive the tribe is in borderline situations. Zero (0) is the median value, -1 means shoot first and talk later, and 1 encourages negotiation. No other values are allowed; do not put any other value in this field.
1 1	Expand	Expansionism versus development is what this field controls. Zero (0) is the median value, -1 enables the tribe to pay some attention to city and terrain improvements, and 1 generally leads to lots of little cities peppered around the map. No other values are allowed; do not put any other value in this field.
1 1	Civilized	This setting controls the relative militarism of the civilization, in terms of research goals. In conjunction with the Civilized Modifier for each advance, this helps guide the AI's research decisions. Zero (0) is the median value, -1 makes the tribe more interested in military R&D, and 1 steers the tribe toward less violent pursuits. No other values are allowed; do not put any other value in this field.



1,	Government	You can override the default titles by setting tribe-specific titles for each of the types of government. This field denotes the type of government in question. The seven possible types correspond to the “levels” of government. They are: Anarchy (0), Despotism (1), Monarchy (2), Communism (3), Fundamentalism (4), Republic (5), and Democracy (6). The next two fields define the new titles.
Dictator,	Male Title	This is the title used for a male leader under the specified type of government. If you set the Gender toggle for this Tribe to zero (0), this title is used under that type whenever the AI controls the civilization from the start of the game.
Dictator,	Female Title	This is the title used for a female leader under the specified type of government. If you set the Gender toggle for this Tribe to 1, this title is used under that type whenever the AI controls the civilization from the start of the game.

NOTE

The different titles set in *rules.txt* don't always work. A better bet is to change the titles in *game.txt*. That method, however, is not tribe-specific.

The last three fields are optional, and can be repeated up to seven times—once for each type of government. In the Roman example, there are two repeats; the first is for Despotism (1), and the second is for Monarchy (2). Put a comma after the last field (Female Title) if you want to add another pair of titles.

Table 7-2. Tribe Colors

NUMBER	COLOR
1	White
2	Green
3	Dark Blue
4	Yellow
5	Light Blue
6	Orange
7	Purple

Other Things in *Rules.txt*

While you're in the *rules.txt* file, there are a few things you can change that haven't been mentioned yet—primarily because they don't fit any of the categories discussed so far. They're all minor, but if you're serious about getting every little detail of your scenario just right, you'll want to know about these.

@GOVERNMENTS is the header for the section that defines the default titles for each type of government. You might want to change the names of the government types, too, so that the window in which the player can customize the titles shows the right names.

@CARAVAN is header line for the list of the possible commodities that Caravan and Freight units (or your renamed equivalents) can carry. These names are also reflected in all the Supply and Demand features.

@ORDERS defines the little letters that mark the shields of units performing specific orders. For example, a fortified unit has an "F" on its shield. Note that you do not change the names of the orders on the Orders menu here; if you change the names of the orders here, it shows up in the Status box.

@DIFFICULTY marks the beginning of the names of the Difficulty Levels. Editing this does not change the selection box that appears when you start a new game. Rather, these names show up whenever a player opens a saved game or makes it onto the high scores list.

@ATTITUDES heads up the list of adjectives used to describe enemy rulers' disposition when the player engages in diplomacy with them. Changing these can have amusing results.

The Game Setup

You've done a lot of preparatory work so far, and if you've done it right, the actual process of creating the scenario should be a breeze. A *Civ II* scenario is really just an alternative version of a saved game file—with some extra data in it. Therefore, to create a scenario, you need to start a game of *Civilization II*.

When you start the new game that will become your scenario, those little decisions you make at the start of every *Civ II* game—the game setup—are extremely important. Once you're in the scenario, you can undo or edit almost everything you do there, but you only get one shot at some of these choices. If you change your mind later, you'll be forced to remake your scenario from the beginning—from this point—and that's a pain.

Your First Decision

The first step in the process of starting the game that will become your scenario

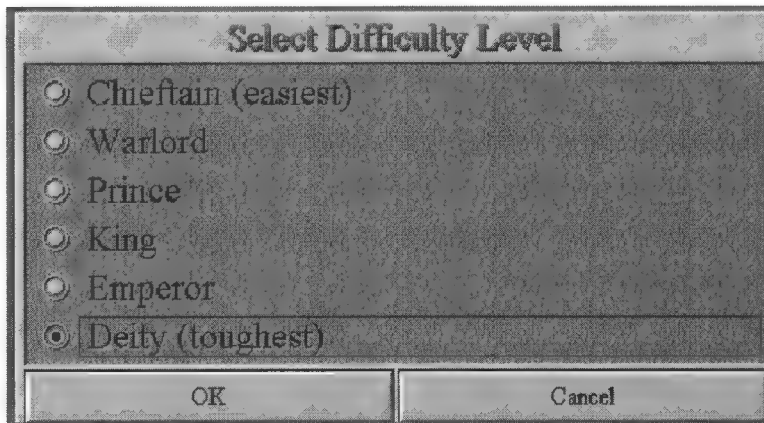
is an easy one. All you need to decide is what map you want to use (for more details on using these options, refer to Step 2: Create Your Map).

- ▼ **Start a New Game** if you don't really care what map you end up with.
- ▼ **Start on Premade World** to select and use a map that you have created or borrowed.
- ▼ **Customize World** is for those of you who want some control over your map, but don't want to do all the work of creating your own.

- ▼ **Begin Scenario** if you're planning on borrowing not only the map from an existing scenario, but some of the setup, too. This is also how you load the scenario file version of a partially completed scenario.
- ▼ **Load a Game** is the option for you if you're building your scenario using an existing game as your base. This is also how you load the parallel save version of a partially completed scenario (for more details see the section on The Importance of Save Files later in this chapter).

Difficulty Level

What you choose here becomes the default Difficulty Level for your scenario. The player can select a different difficulty, so set this at whatever level you think will be the most fun.

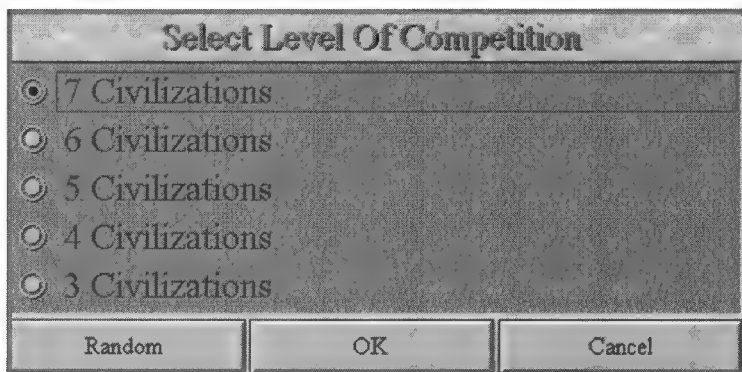


I suggest you always use Deity level. That way, while you're setting up the scenario, you can arrange the cities for the happiness situation you want, and you'll know that players have the power to make things easier on themselves.

Note that what you choose here has its usual effect on the player's Civilization Score.

Level of Competition

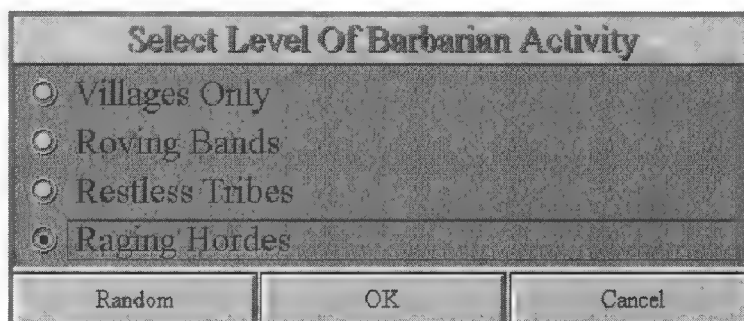
When you decide how many civilizations you want in your scenario, make sure you get it right the first time—or if you're not sure, you can choose a higher number than you think you'll need. Once you've started the game and begun work on the scenario, you can't change your mind and raise this number without re-starting the game and redoing all the setup work you've done.



If you set the number too high, however, you can always use the Kill Civilization option on the Cheat menu to nix the spare tribes. (Sometimes they come back a few times, but sooner or later they stay gone.) The problem with this, though, is that those killed civilizations show up in some of the game history displays, like the Power Graph and the guillotine animation. Your best tactic is to choose correctly in the first place.

Barbarian Activity

The question for most scenarios is not how often barbarians will show up, but whether they are appropriate at all. If you've set up a detailed, perfectly balanced situation—especially an historical re-creation—you probably don't want barbarians popping up and potentially spoiling everything.



To remove barbarians from your scenario entirely, choose Villages Only at this menu, then later use the Wipe All Goody Boxes option in the Scenario Parameters (on the Cheat menu) to remove the villages.

If you do decide to allow barbarians in your scenario, choose the level you believe will make the game the most fun—or the level that best fits the needs of your scenario.

Note that what you choose here has its usual effect on the player's Civilization Score.

This is a decision that you cannot undo or change in any way once you have created a game and begun work on the scenario. If you change your mind about the level of barbarian activity you want in your scenario, you will have no alternative but to start over from this point.

CAUTION

Game Rules

When you play a game of *Civilization II*, you probably use the Standard Rules most of the time. I know I do—it's easier. When you're building a scenario, however, you'll almost certainly want to choose Customize Rules. Here's why:

- ▼ **Simplified Combat** is not normally something you'll want to use. However, scenarios are as varied as people; it might be appropriate.
- ▼ **Flat World** gives you an opportunity to change your mind about whether the world in your scenario is flat or round. Note that what you decide here overrides whatever setting you defined using the Set World Shape option in the Map Editor.
- ▼ **Select Computer Opponents** is one option that you will almost certainly want to use—unless all of the tribes you defined in *rules.txt* are interchangeable. You can select one tribe of each color, up to the number you chose in Level of Competition. Later, when you get around to selecting a tribe to rule, your choices will be limited to the ones you pick here. Note that if you have less than seven tribes in the scenario, you might not get the choices you expect. Sometimes you need to start the game more than once—until you get the tribes you need.
- ▼ **Accelerated Startup** is a bit troublesome. On the one hand, it seems like it might be a useful shortcut when building scenarios that take place in modern times. On the other hand, you never know exactly what you're going to get, and you could spend more time fixing things than you would have setting them up from scratch. I don't use this, but if it works for you, by all means use it.

- ▼ **Bloodlust** is applicable to a surprising number of scenarios. The potential for building spaceships just isn't appropriate in most situations. Leave this unchecked only if you want civilizations to be able to build space ships bound for Alpha Centauri, or if you have made the relevant wonder, the necessary improvements, or the required portions of the advance tree completely inaccessible.
- ▼ **Don't Restart Eliminated Players** can be handy, especially in historical scenarios. For example, after the player, ruling the British, defeats the Spanish Armada and conquers the Spanish empire, you certainly wouldn't want a tribe of Aztecs popping up somewhere in Scotland. However, if you're building nearly an entire new game, this can inject an extra complication for your players.

CAUTION

If you customize the rules, remember that none of these decisions can be undone or changed in any way once you have created a game and begun work on the scenario. If you change your mind about any of these settings, you will have no alternative but to start over from this point.

TIP

I mentioned in Step 2: Create Your Map that *Civilization II* inserts polar caps onto a map when you load it for the first time. You can avoid that particular feature by selecting Customize Rules. Even if you don't want to use custom rules, and even if you leave all the checkboxes unchecked (thereby using the default rules, anyway), just going through the motions prevents the *Civ II* program from putting in those annoying polar caps.

Gender, Tribe, and the Rest

Whenever a player loads your scenario, he or she picks a gender and a tribe to play and has the opportunity to change the ruler's name for that tribe. Also, if you've been following my advice (in Defining the Tribes), you have determined what you want all of these to be for each tribe by editing *rules.txt*. So, what's the point of changing any of this? There isn't much point.

When you begin building your scenario, the only decision you need to make on any of these five points is to choose a tribe and a city style. Note that, since you must choose a gender *before* you pick a tribe, you must make sure to select the gender that matches what you defined for the tribe you intend to choose. At all the other menus, you should just accept the defaults that you yourself defined. One odd side-effect you should be aware of is that whatever city style you pick now, the player's cities will be displayed in that style—no matter what is set in *rules.txt* for the tribe the player chooses to rule.

Which tribe you pick depends on only one thing: which one is first in the list. If you choose any other tribe, the civilizations listed before that one will take their turns before you get control of the game. That means that the computer AI will be *doing things to your scenario*—building cities, making decisions, and generally mucking everything up. You don't want that. Pick the first tribe in the list.

If, for any reason, the names of kings are not right, you do not need to start over. You can correct the names using the Edit King option on the Cheat menu.

NOTE

The Importance of Save Files

Once you've gotten this far, you don't want to lose the work you've already done. You could protect your progress using the Save as Scenario feature—and you should—but you should *also* save the scenario as a regular save file. This is known as keeping a *parallel* save. If you don't keep parallel save files at each important point in the process, you can find yourself redoing an awful lot of work.

Why? Two reasons, one of which is more complex than the other. The simple reason is that your scenario file always needs to have the same name. Thus, saving multiple versions of it can be a chore—all the renaming and copying and so forth.

The more complicated—and more important—reason involves the differences between the two types of file. Some information that's kept in the scenario file—things such as which civilization's viewpoint each civilization uses—is not saved in the normal save file. If you somehow make a mistake involving one of these scenario-specific features, reverting to your last scenario save might require redoing a lot of work. That would suck. However, if you save the scenario—even after you've made the mistake—as a normal save file, the specific information gets lost, and when you load the saved game, those things revert to their default settings.

That's not too clear, so let's take an example from real life. While working on the Mars scenario, I placed all the units for several civilizations during the same editing session. First, I did the Americans. Then I used Set Human Player to change over to the Japanese and place their units. However, as I had not revealed the entire map, I couldn't see what I was doing—the American view persisted. So I used the Reveal Map feature to see the Japanese view. That worked, so I happily did the same every time I switched civilizations. However, after I saved the scenario and went back and tried to play it (to test the newly placed units), I found that every civilization had the wrong view! I had somehow assigned another civilization's viewpoint to every tribe in the scenario.

I had not been keeping parallel saves, so I had to go back and re-place all of those units by hand—and redo everything else I had done during that session. What a pain. If I had had a normal save file, however, that file would not have saved the viewpoint assignments. When I loaded the (imaginary) normal save, all of the civilization's viewpoints would have been set to their defaults—their own viewpoint.

Save early and often.

Building Empires

Having begun the game and (if you're smart) saved it twice, now you'll want to get going on the actual setup. How little or how much work this entails depends entirely on your design. For example, if you've played the Age of Reptiles scenario, you know that there wasn't a lot of setup work involved in creating the situation; every civilization in that scenario started with nearly nothing. At the other extreme, the World War '79 scenario posits a complex, pre-existing population, industrial, military, and scientific base for every civilization involved. That took some doing.

Generally speaking, the building of a scenario goes most smoothly if you proceed more or less in the order outlined in this section. As always, however, if something else works better for you, do it that way.

Establishing the Capitals

Assuming that you listened to me and picked the first tribe in the list when you started up the game, you're in control of the civilization that takes its turn first (see Figure 7-1). That's extremely important, because it enables you to determine where every civilization's capital city is.



Figure 7-1: Nothing has happened yet.

Your first action should be to go to the Game menu, choose Game Options, and turn Always Wait at End of Turn *on*. This helps prevent the game from getting out of your control, because no turn can end until you take an action to end it—pressing Enter, clicking in the Status box, or changing which civilization you control. After the capital cities are established, you can pretty much undo anything that you accidentally allow the AI to do, but it's better to prevent the problem in the first place.

Your second action should be to activate the Cheat mode, then use the Reveal Map option on the Cheat menu to make the Entire Map visible. That way, you can see everything, without affecting the viewpoint of any of the civilizations.

With *Fantastic Worlds*, you can use the Toggle Scenario Flag option at the top of the Editors menu to activate Cheat mode, create your scenario directory, and save your new game as a scenario file—all in one fell swoop. Personally, I find this method to be less desirable and more complicated than doing it all manually, because it assumes that you haven't done any of the preparatory work that I've described so far.

Do what works for you; it could be that I'm just set in my ways.

**FANTASTIC
WORLDS**



CAUTION

Under no circumstances should you use any of the tribe-specific Reveal Map options while building a scenario. If you don't know why, go back and read the section The Importance of Save Files.

If you set starting positions when you created your map, there's a Settler (or equivalent) standing in every one of the places you selected. (For the rest of this discussion, I'll just refer to it as a Settler; you know what I mean.) The Map window is centered on the one you're currently controlling. Nothing says that you must establish the civilization's capital here, but if that's what you want, go right ahead and press B. If you'd rather put the capital somewhere else:

- 1 Right-click on the spot you've selected (to get the map cursor there).
- 2 Use the **Create Unit** feature (on the **Cheat** menu) to put a Settler there.
- 3 Click on the new Settler to activate it. (Newly created units have a full movement allowance.)
- 4 Press **B** to establish the city.
- 5 Name it. If you edited the list of city names in *city.txt*, you should be happy with the default name. After all, you created it.

That takes care of the first civilization. Now you need to change control to another one without the AI getting control of anything in the meantime. First of all, if you created a new Settler, go back and use the **Destroy All Units at Cursor** option (on the **Cheat** menu) to get rid of the original one. It should now be the end of the first civilization's turn, and the game is waiting for you to press Enter. (If it's not, there's another unit roaming around somewhere; find it and destroy it.)

Save the game if you want to, then use the **Set Human Player** option on the **Cheat** menu. The list that appears has the civilization that you're currently running already selected. This list is generally in the same order as the sequence of turns, and as soon as you pick a new civilization to rule, the AI instantly takes all the turns in between the one currently in progress and the next turn of the civilization you pick. Therefore, if you choose the next civilization in the list (immediately after the one you're running now), the only action the AI can take is to officially end the first civilization's turn. Then it's your turn again. (If this tactic fails, don't get upset. Everything the AI might do, you can undo).

Now repeat the procedure for establishing a capital, make sure it's the end of the turn, and move on to the third civilization in the same way. Continue like this until



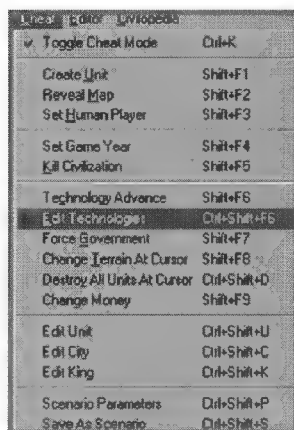
you have established capital cities for all of the civilizations in your scenario. When you've finished that, *save the game*.

The State of Science

As soon as you've put down all of the capital cities, it's time to decide who already has what advances. This might seem a little premature to some of you, so let me explain why. There are a few major reasons—and one caveat you should consider.

- ▼ First and foremost, you cannot put improvements and wonders into cities until the prerequisite advances have been “discovered” by the civilization that owns the city. You can't cheat to generate advanced buildings like you can units.
- ▼ For that matter, having the advances defined is usually more convenient when you're creating all the units, too. The only time it's an inconvenience is when you're putting down obsolete or irreplaceable units.
- ▼ When you get around to making terrain improvements (a little further on) and trying to balance every city's size, resources, and improvements with its happiness and support needs, it's helpful to have everything as it is meant to be—obsolete wonders are obsolete, governments are in place, advances' fixed effects are already in effect, and so on.

You can probably think of several reasons not to assign advances this early in the process (for instance, maybe you want a city to be *building* obsolete units when the scenario begins), but there's a way around all of them. Simply take the troublesome advance away again—temporarily—while you do whatever it is you need to do. Trust me, the overall convenience outweighs the exceptions in almost every case.



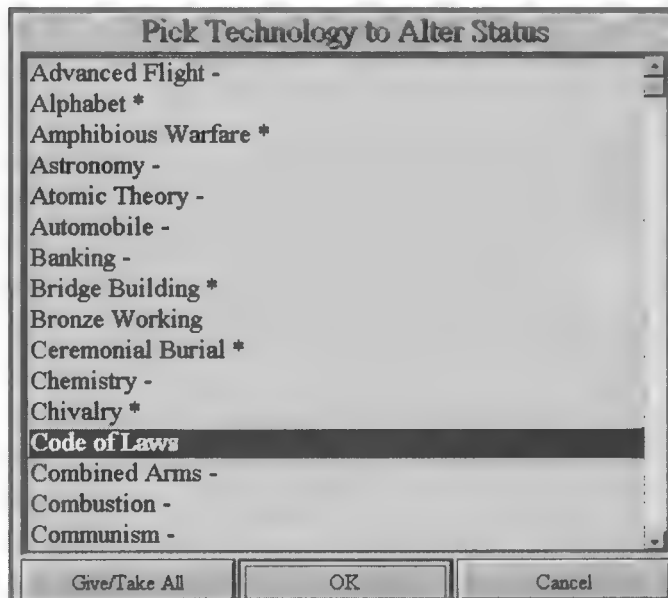
Linear Editor	Unlabeled
Toggle Cheat Mode	Ctrl+K
Create Unit	Shift+F1
Reveal Map	Shift+F2
Set Human Player	Shift+F3
Set Game Year	Shift+F4
Kill Civilization	Shift+F5
Technology Advance	Shift+F6
Edit Technology	Ctrl+Shift+F6
Force Government	Shift+F7
Change Terrain At Cursor	Shift+F8
Destroy All Units At Cursor	Ctrl+Shift+D
Change Money	Shift+F9
Edit Unit	Ctrl+Shift+U
Edit City	Ctrl+Shift+C
Edit King	Ctrl+Shift+K
Scenario Parameters	Ctrl+Shift+P
Save As Scenario	Ctrl+Shift+S

Keep in mind when doling out advances that who has what technologies is vital to the balance of a scenario. You set everything up in *rules.txt*, so you know exactly what each advance makes possible. Don't give them out recklessly.

Happily, you do not need to spend any game time (turns or portions thereof) giving advances. You can assign advances to every civilization, regardless of which one you happen to be leading.

- 1 Select the **Edit Technologies** option from the **Cheat** menu (*not* the Technology Advance option).
- 2 Pick the first civilization in the list.
- 3 Give and take advances by selecting each and clicking the **OK** button (see Figure 7-2). Continue until every one you want that civilization to have is marked with an asterisk. (Those the civilization is able to research immediately are marked with a minus sign.)
- 4 Click the **Cancel** button when you're finished. (I know it doesn't make sense, but that's how it works.)
- 5 Now do the same thing for every other civilization in the scenario, then save the game. Note that the **Cheat** menu feature **Edit King** has an option called **Copy Another King's Tech** that might be a helpful shortcut.

Figure 7-2: This is your tool for editing technologies.



The other potential pitfall to watch out for is the spread of technologies. There might be some advances—the ones leading to private tech trees, for example—that you've

given to specific civilizations, and (for whatever reasons) you don't want those to pass to other tribes. Sometimes it's more a matter of *when* the technology is communicated; you'd prefer that science be kept private in the beginning of the scenario. There are four strategies for slowing or preventing the spread of technologies among the civilizations in your scenario:

- 1 Put distance between the civilizations. If they can't reach each other, they can't trade, steal, or conquer advances. This is a slowing tactic, not a preventative one.
- 2 You can use the scenario parameter **Forbid Tech from Conquest** (described later in this chapter) to prevent captured cities from giving up advances.
- 3 Use events to prevent tribes from negotiating and trading advances (for the details, read Step 8: Schedule Events).
- 4 To put the last nail in the coffin, make sure that there are no diplomatic units in the scenario to steal advances—at least not until so late in the tech tree that it won't make any real difference.

Research Goals

Now we get to the caveat I mentioned. During the first turn of the scenario, most players expect to be able to choose the next research project (in game terms called the “research goal”) for their civilization. Unless you intend to take that choice away, you'll need to do some last-minute fiddling.

In the process of creating the scenario—laying down cities, placing units, and all the other little jobs—you might go through several game turns. During those turns, the chances are good that you'll be prompted to choose the next research goal for every civilization in the game. You can, of course, reset the year and determine the exact starting date for the scenario; but if you want players to have their choice of research goals, there's an extra step you must take just before you save the game as a scenario for the very last time:

- 1 Select the **Edit King** option from the **Cheat** menu.
- 2 Pick the first civilization in the list.
- 3 Use the **Clear Research Goal** option.
- 4 Now do the same thing for every other civilization in the scenario.

Tech Paradigm

The last consideration in the scientific realm is the speed at which advances come. Be careful when lowering this number—it's amazing how much less interesting the game gets when there's nothing left to research but Future Tech. Raising the paradigm has its dangers, too. Unless you're making progress tough intentionally—to keep a historical scenario from researching itself right out of its milieu, for instance—try to limit yourself to small increases. Remember, *Civilization II* is a well-balanced game to begin with. Any change you make should be for a good reason.

Also keep in mind that the player has many methods of raising and lowering the research output of a civilization—city and terrain improvements, the Tax Rate, and so on. Don't assume that just because you can't get around the new Tech Paradigm, no one can; you might be surprised what an enterprising and dedicated *Civ II* player can do.

NOTE

If you want to eliminate science from your scenario entirely, the best way to do so is through an excessively high Tech Paradigm—50 or 60 should do. Remember, you can't cut off access to Future Technology, so there's no way to make a scenario with absolutely no research. If the paradigm is high enough, however, the difference is unnoticeable.

To set the Tech Paradigm for your scenario, simply:

- 1 Select **Scenario Parameters** from the **Cheat** menu.
- 2 Choose the **Tech Paradigm** option.
- 3 Enter a number.
- 4 Save the game.

TIP

You can change the Tech Paradigm in *rules.txt*, too, but don't bother. The setting you enter in the scenario parameters supersedes anything you put in the Cosmic Principles.

Creating Cities

Once the capital has been established, you can move on to create all the other cities of each empire. The reason I stuck science in between the two city steps will become clear when we start turning the bare, size 1 cities we're creating into real ones—making them bigger and putting in improvements, units, and wonders.

The process for creating other cities is exactly the same one you used to create the first city, except that you don't need to move from civilization to civilization after each city. (You only did that to ensure that no mistakes got made, since repositioning a capital is troublesome.) Build all the cities for one civilization, then save and go on to the next one.

There are a few simple guidelines you should remember when placing cities.

- ▼ **You can't put two cities in adjacent squares**, not even if you want to. The game program won't allow it.
- ▼ **Don't overlap city radii**. It isn't always possible, but when you have the choice, don't place cities so that the city radius of one contains a portion of the city radius of another. Turn on the map grid if you're not sure.
- ▼ **You can't build cities at sea**. So don't even try.
- ▼ **Consider the terrain**. Just as you would if you were playing the game, think about the production capacity of the place where you're building a city—especially the City Square, but also the entire city radius. How big can the city reasonably be expected to grow? Will it be able to sustain itself at all? Is the area defensible? Take your intentions for the city into account, too.
- ▼ **Keep clear of the poles**. Any city built too near the top or bottom of the map has only a partial city radius to draw on for resources (see Figure 7-3). Unless you're intentionally limiting the city's chances, stay at least two squares from the edge.
- ▼ **Rivers are roads**. If a city is on or adjacent to a river, travel both to and from the city is eased. Among other things, this means that enemy units can approach quickly from outside the city radius.
- ▼ **The AI never uproots an existing city**. Once you've placed a city, it's there for the duration of the game unless it's destroyed (through combat or starvation) or a human player chooses to uproot it.
- ▼ **Consider your events**. I know I haven't discussed events yet, but just bear with me. If you have events planned for specific regions, you must consider the effects those events might have on the cities' whereabouts.

- ▼ **Allow ports.** If you want a civilization to launch a navy, make sure that at least one of its cities is on a coastline. Avoid putting cities on the coasts of small inland seas, unless you really want civilizations wasting their resources building useless boats.

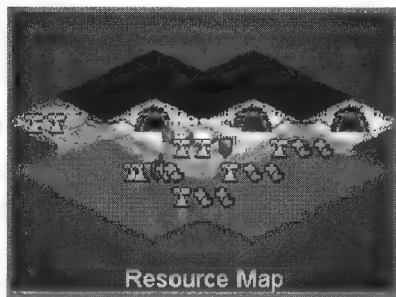


Figure 7-3: *Don't build too close to the edge of the world.*

As you're going through all this setup, you might notice that game turns are elapsing. (If you're very careful, they won't.) Don't let that worry you. You're going to set the exact starting year of the scenario—*after* you get all the pieces of the situation into place. You can then use the Cheat menu options—Edit King and Edit City primarily—to fix the accumulated resources and money.

Size and Contents

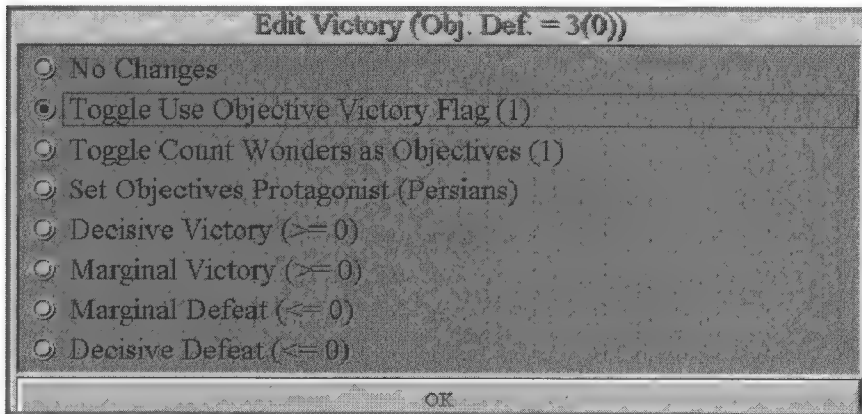
Once the cities exist, you can turn them into more than empty, size-one hamlets. It's not necessary to put *all* the cities into place before you start filling them, but it doesn't hurt. (I'm sure you've noticed that, when it comes to scenarios, I like to take things one step at a time. It helps me to avoid mistakes.) Go through this process for every city:

- 1 Set the size of the city. Right-click to place the map cursor on the city, then pick **Edit City** from the **Cheat** menu. Choose **Change Size**, then enter the number of citizens you want there.
- 2 Return to the normal view and open the **City Display**. Put everyone to work, and—for the moment—don't worry about production, happiness, or any of that. (That comes much later, while you're improving terrain.)
- 3 Next, create the units you want starting the scenario in this city. Note that these are not the units *supported* by this city and they're not units that will be *near* the city. These are only the units you want *inside* the city at the beginning of the scenario. Click the **Change** button in the **Production** box. Choose a unit from the list and click the **Cheat** button. Continue until all of the units you want are in the city.

- 4 If you want any units in this city that do not appear in the production list, don't worry. If you close the **City Display** but leave the map cursor on the city, you can use the **Create Unit** feature on the **Cheat** menu to place obsolete and advanced units in the city square, which means they're in the city. (In fact, you can create all the units in the city this way if you prefer.)
- 5 Use the same **Production** box method to put the necessary city improvements and Wonders of the World in the city. There's also an extremely handy feature in the **Edit City** option on the **Cheat** menu that enables you to copy all the improvements another city has.
- 6 Before you leave the **City Display**, you can set what this city is building. Players can change this, so it's not necessary. However, if it fits your scenario or is necessary (perhaps one point of the plot is that a certain city has something powerful half built already), make sure to assign a project.
- 7 Exit from the **City Display** and save the game.

Objectives and Victory Conditions

Making your scenario to be objective-based, rather than relying on the typical *Civ II* rules for winning and losing, adds a step to your scenario-building job. It's not difficult, but like a lot of the game design process, it takes a little time and work. Why would you choose the objectives system over the normal one? The objectives are most often appropriate for historical scenarios, when capturing specific territory, not the entire world, is a better measure of victory.



First of all, you must decide which cities you want to be objective cities. Typically, you'll choose cities that were important in the historical situation you're re-creating

or which are key in whatever situation you're setting up. In some scenarios, every city should be an objective. Some particular urban centers are much more important than others. These, you can make major objectives, which are worth three times as much. Note that any city built after the game begins can never be an objective.

Here's what you do to set up objectives for your scenario:

- 1 Choose your objectives. To make a city an objective (or major objective), select the **Edit City** feature on the **Cheat** menu. Near the bottom of the list of options are the two options for creating objectives. Select the one you want.
- 2 Next, pick the **Scenario Parameter** options on the **Cheat** menu. Select **Edit Victory Conditions** from the menu that appears.
- 3 Choose the **Toggle Use Objective Victory Flag** option and click OK. This tells *Civilization II* that your scenario is an objectives scenario.
- 4 Decide whether Wonders of the World will also count as objectives in your scenario. This matters, because if you make wonders into objectives, it becomes impossible for any civilization to build new wonders during the game. If the answer is yes, pick **Toggle Count Wonders as Objectives** and click OK.
- 5 Often, the toughest part of this is deciding which tribe is to be the protagonist nation. Whichever it is, all the other civilizations are going to be ganging up against it. (Everyone else's success or failure is measured against that of the protagonist.) Make sure this is neither the weakest nation on the planet, nor a civilization so powerful that none can stand against it.
- 6 Last and most complex are the four measures of victory. First of all, notice that the title of the **Edit Victory Conditions** window includes two numbers. The one in parentheses is the number of objective points currently controlled by the protagonist nation. The other number is the total number of objective points in the scenario. Let these guide your decisions here. Otherwise, what these numbers should be is entirely dependent on your scenario.

New City Icons

The city icons in the original game are very nice, but they won't be appropriate for every scenario. Luckily, you can change them. It's not difficult to make new icons, but you need a way to edit GIF files.

If you have *Fantastic Worlds*, you can modify the looks of every city unit icon using the Icon Editor in the Cities Editor on the Editors menu. This method is more convenient and easier than a GIF editor for most folks.

NOTE

For more info on the GIF format, read the sidebar titled “GIF Editors” in Step 1: Getting Started.

The remainder of this section assumes that you have a GIF editor or *Fantastic Worlds*.

All the city icons (and a few other useful things) are contained in the file *cities.gif* (see Figure 7-4). Once you understand the way this file is organized and how to find the icon you’re after, you should have no trouble editing the looks of all six city styles.

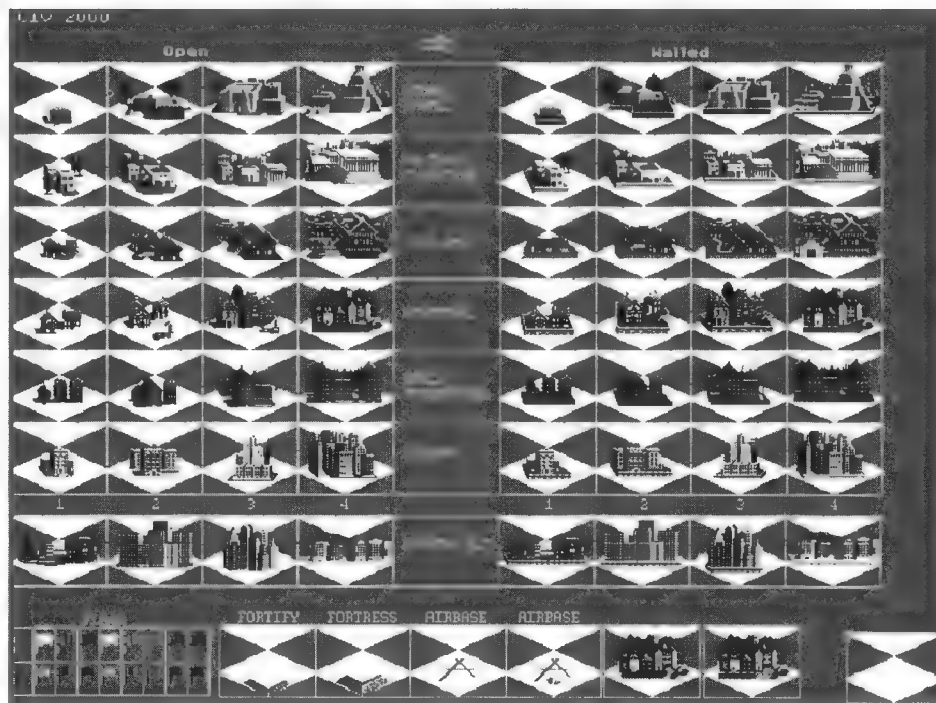


Figure 7-4: *Cities.gif* contains the city icons, and more.

The Palette and Green Lines

Files in GIF format are one example of what are called *paletted* graphics files. The palette (sometimes called a “color table”) is a collection of colors defined for use in the file, and no other colors appear in that file. Most graphics programs include a way to display the palette, usually as a box full of colors you can pick from. All of the files for *Civilization II* use the same palette, and you should never change the palette in any of the files—ugly graphic problems result, and the game could crash, too.

The *Civ II* palette contains some colors that are recognized by the program as “not to be displayed” colors. These are mostly “transparent” colors that we can use in the files to help us position graphic elements correctly. For example, the city icons all sit on a purplish background in the shape of a normal terrain square (also called a “tile”). That purplish diamond sits on a grey background. Both the purplish color and that exact shade of grey are transparent colors. When the icon is displayed on terrain, you can still see the terrain underneath—in every place that the purplish color and the grey appear in the graphic file. The purple tile gives us a way to know where on the terrain tile the icon will appear.

CAUTION

Some colors in the palette are very similar to the transparent colors, especially the transparent shade of grey. If you get them mixed up, your game will look pretty bad, but probably won't crash. Just re-edit the file and replace the wrong color with the right one, and the problem goes away.

City icons can overlap the upper gray areas, but should *never* cross over the lower borders of the purple diamond.

Another color that you must be careful about is the bright green used to draw all the boxes around the icons. Those green boxes are used by the program as reference to tell where the icons are. Every city icon is in a box of exactly the right size and shape, and you should never mess with the green lines. Work inside the lines.

CAUTION

Never move or erase any of the green lines. Don't add new ones, either.

Active Areas

If you look in the *cities.gif* file, you'll notice labels that obviously aren't cities and a few icons that weren't used in the game. The artists left some junk in the file, plus

there are some bits that the program uses. Anything that does not appear in the game is in an *inactive area* of the GIF file. Anything outside of the *active areas* is either ignored by the game program or is something you should never touch (see Figure 7-5), so there's no point in changing any of it.

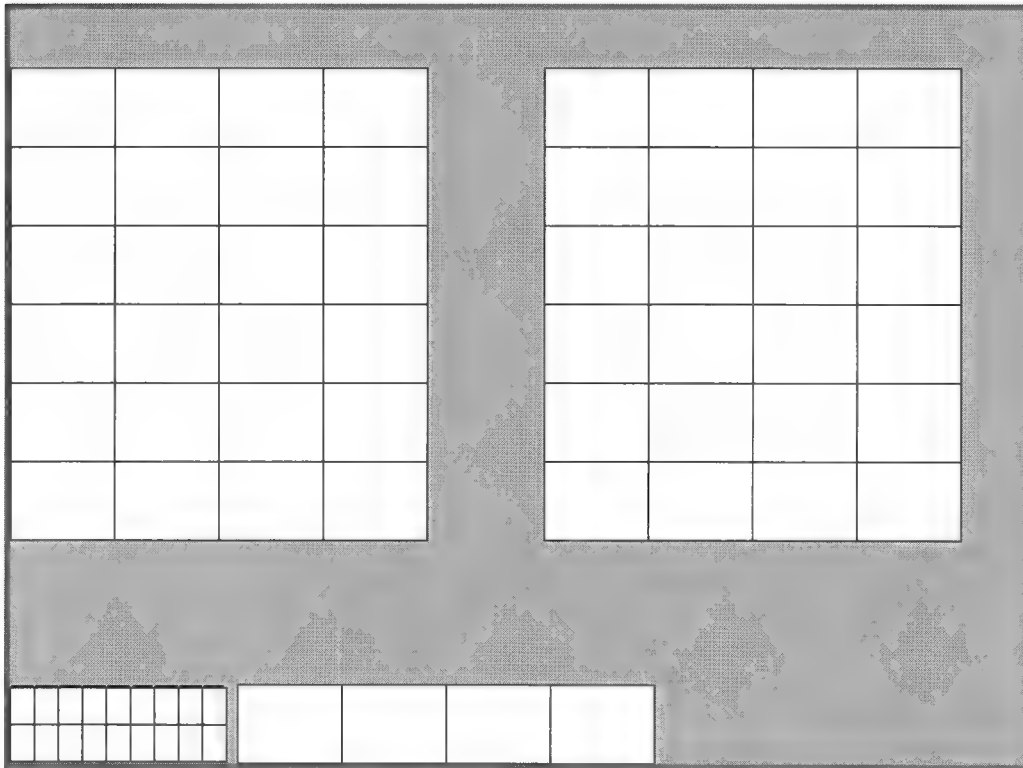


Figure 7-5:
Active areas of
Cities.gif

Styles, Sizes, and Walls

Changing a city icon is a simple matter of modifying the picture that matches the style and size. The exact location of the city icons is noted in Figure 7-6.

Figure 7-6: Where the city icons are

OPEN					WALLED			
Small	Medium	Large	Extra Large		Small	Medium	Large	Extra Large
				Stone Age				
				Classical				
				Far Eastern				
				Medieval				
				Industrial				
				Modern				

Each row of icons corresponds to a single city style, from 0 at the top down to 3, followed by the *Industrial* and *Modern* styles. The icons in the left half are the Open cities, while those on the right half are the Walled versions of the same cities. Each column represents a city size—from Small at the left to Extra Large on the right. The combination of Style, Size, and wall status determines, for each city, which icon appears in the game during any given turn. Thus, for example, the icon in row 1, column 3 on the left appears in the game for all large, open, bronze age-style cities.

The Flag and Disorder Pixels

There is one (and only one) circumstance in which you should mess with the green lines that define the city icon boxes. That's when you place the *flag pixels* and the *disorder pixels*.

If you look closely at the green lines bordering each icon, you'll see that there are two blue dots and two orange dots associated with each city icon. One of each is somewhere in the green line that defines the top of the unit box, and the others are in the green line along the left side. The orange pair, taken together, tell *Civilization II* where to place the flag that flies over the city when it is occupied by at least one unit. Similarly, the blue pair tell the program where to place the civil disorder icon (when the city is in disorder).

Regardless of which of these graphic elements we're dealing with, the pixel on the top defines the leftmost edge of the element. (For this reason, it's sometimes called the "left pixel," but that gets confusing.) Make sure that this point is far enough from the right side of the box, or the element will overlap the box and cause leftovers—the overlapping portion is not erased—when the flag or icon is removed. For the same reason, never put this pixel in the leftmost 2 pixels of the box, either.

The pixel on the left defines the top edge of the element. (Therefore, it's sometimes called the "top pixel," so you can confuse it with the other one.) Make sure that this point is far enough from the bottom of the box, or, again, the element will overlap the box. For the same reason, never put this pixel in the upper 2 pixels of the box, either.

Whenever you change the look of a city icon, you should determine where the flag and disorder icon will look best, then use these pixels to put them in their places.

The Icon Editor in the Cities Editor does not have any tool for placing the flags and disorder pixels. You're on your own for that.



If there are no flag pixels or no disorder pixels associated with a unit, the program places the element in question into the upper left-hand corner of the box, and when you play the game, that unit will leave leftovers (the unerased overlapping bits).

Do Not Touch

Here's a general rule that I apply to all the *Civ II* files: If you don't know what it is and how it works, don't touch it.

When it comes to the cities file, there are only a few parts that neither you nor I nor anyone else should change: the ones we've already mentioned—the green lines, the purple tiles, and the lower grey areas. You can mess around with the flags, but I don't recommend changing the major color of any of them.

Governments

The next step is to give each of the empires in your scenario a form of government. Now that the cities are in place and built up with whatever units, improvements, and wonders are appropriate, it's time to start thinking about the resources and output

of the cities. The major effects of a civilization's form of government are on production, food, and trade. Therefore, it's helpful to have the government in place before the final step in the process of balancing the cities—modifying the surrounding terrain.

The production effects are important, but don't lose sight of the other ramifications of each form of government. Unanticipated side-effects can undo an otherwise interesting scenario. For example, if you plan for a particular empire to be warlike and aggressive, you shouldn't give that civilization a Republic or Democracy as a government—unless you activate the Total War option, that is (see the section The Telling Details, later in this chapter, for the details).

Assigning a government to one of the civilizations in your scenario is not difficult.

- 1 Select **Force Government** from the **Cheat** menu.
- 2 Choose the civilization you want to work on.
- 3 Pick the type of government you want that civilization to begin the scenario under.

TIP

At this point, you are only specifying the form of government that each empire will start with. There's no guarantee that any of them will stick with the type you've assigned them. If it's important to you (or to the scenario) that tribes not switch governments, use the **Forbid Government Switching** option in the **Scenario Parameters**.

Modifying Terrain

The Map Editor only enables you to do so much. Now that you're in the scenario, you can finish the job. All you have so far are nude cities. Each one needs a certain amount of support infrastructure—improvements to the surrounding terrain. You'll want roads or railroads to run between some locations, too, and maybe a few strategically placed fortresses and air bases. As always, exactly what you need depends on the design of your scenario.

It makes no difference which civilization you're in control of when you make changes to the terrain. What matters is that you have the Cheat mode active and the Entire Map revealed. This enables you to look inside the cities of all civilizations, so that you can gauge what each needs in terms of supporting infrastructure in the city radius—irrigation, roads, mining, railroads, and farmland.

When placing terrain improvements, keep all of the effects in mind. Sometimes, you might need to sacrifice accuracy for game balance. For example, railroads not only add production, but also allow free movement. The original design for the American Civil War scenario called for simulation of the rail networks that were so important during the actual war. Unfortunately, the ease of movement this made possible enabled either side (whichever side the player ruled, usually) to win in only a few turns. The railroads were downgraded to roads, which better simulated the historical *effects* of the rail networks.

TIP

Assuming that you have some idea what you need to do, here's how you do it:

- 1 Right-click on your target terrain square to move the map cursor there. (If the map cursor is already active, you can use the keyboard movement controls to move it around just as if it were a unit. This can be really convenient, especially when you're building roads.)
- 2 Use the **Change Terrain at Cursor** option on the **Cheat** menu.
- 3 If you want to change the underlying terrain, click the **Terrain** button and select the new type from the list.
- 4 Otherwise, select any combination of improvements from the list, then click the **OK** button.



Now go back to every city in the world and check the happiness. (I never said this would be easy.) Make sure that each has the setup you want at the start of the scenario—they won't necessarily all be happy cities. Do what you must to fix any incorrect happiness levels. While you're in each city, look over the food, production, and trade levels and

fix what you must. If support is burdensome, don't forget that you can use the Edit Unit feature on the Cheat menu to change the home city of any unit.

You should revisit the cities as you create units, to make sure that the improvements you've made to the terrain (and the improvements inside the city, too) are enough to provide the necessary support for all the units you've assigned to the civilization—or not enough, if you intend for a particular civilization to start off in a difficult situation.

As you progress through the building process, you might come across terrain issues you didn't foresee when you created the map. Change Terrain at Cursor is the perfect tool for tweaking the land to fit your needs—unless you need to move rivers, that is.

NOTE

As I explain in detail in Step 2: Create Your Map, you can't add or remove rivers outside of the Map Editor.

Placing Units

Now that all the underlying structures are in place—cities and their contents, terrain improvements, the government, and so on—we're ready for the active players in the scenario, the units outside the cities.

You can assume that all the civilizations will build units during the game, but there are some units you want in place from the start. How many there are, what type, and where they're situated depend entirely on your design, but there are a few questions you should ask yourself whenever you put a unit in place:

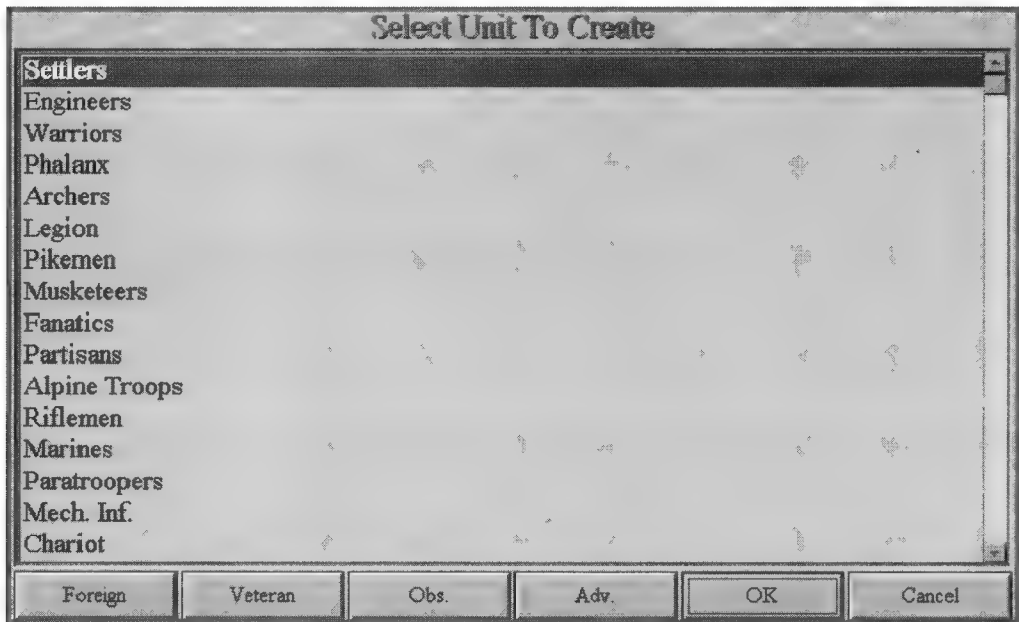
- ▼ **Why this type of unit and why here?** You should have a purpose in mind for every unit—specific reasons why it exists, why it is where it is, and why it's the type of unit it is, rather than some other type.

- ▼ **What will the AI do with this unit here?** Forget to consider this, and you'll cause yourself plenty of problems. Based on the way you've seen the computer play, try to figure out what it will do with this unit—where it can move, what it might do, and what movement of other units its zone of control can prevent. After all, six civilizations out of seven are run by the AI.
- ▼ **What will a player do with this unit here?** The only thing worse than giving the computer a unit it can misuse is giving the same opportunity to a human player. Mick found that out the hard way, when Roscoe (one of our testers) defeated the Confederates and won the American Civil War scenario in three turns.
- ▼ **What unintended effects could this unit have?** This one can be tough to predict. The possibilities range from making a part of the map visible that you don't want explored yet, to building a city or transforming terrain, to establishing an unwanted embassy, to interdicting traffic through a bottleneck point. Think it through.

Assuming that you've been following the procedure as I've laid it out, five things are true right now:

- 1 You're in control of a civilization and it's your turn.
- 2 The Always Wait at End of Turn option is enabled.
- 3 Cheat mode is active.
- 4 You used the Cheat option Reveal Map to make the Entire Map visible.
- 5 You've saved the game a couple of times, both as a scenario and as a regular save file.

If all of that is actually the case, you're ready to begin placing units. The process is simple.



- 1 Right-click on the spot you've selected (to get the map cursor there).
- 2 Activate the **Create Unit** feature from the **Cheat** menu or by pressing **Shift+F1**.
- 3 If you see the type of unit you want in the list, select it. (Note that this is a scrolling list; it might not all be visible at once.) If the unit isn't there, it's either obsolete or advanced—based on the technological state of the civilization you're working on. To add the missing units to the list, click the relevant button (Obs for obsolete and Adv for advanced). Then select the unit.
- 4 Click the new unit to activate it. (Newly created units have a full movement allowance.)
- 5 Choose **Edit Unit** from the **Cheat** menu. If there is more than one unit in the same terrain square, you'll need to specify which one you want to edit.
- 6 Make whatever modifications to this unit you think are appropriate for the situation you're setting up in your scenario. *Veteran Status* makes a unit more powerful in combat. *Set Hit Points* is useful if you want a unit to begin the scenario in a damaged state. *Home City* enables you to decide what city, if any, pays the support (and happiness) costs for this unit. *Fortify* is a handy way of fortifying units instantly.
- 7 Repeat this procedure until you've placed all the units for the current civilization.

That takes care of the first civilization. To move on to the next, use the Foreign button in the unit selection box. (This way, no turns pass.) Once you have selected a tribe, all the units you place thereafter are of that tribe.

Now repeat the procedure for placing units, making sure to save the game in between civilizations. Continue like this until you've placed units for all of the civilizations in your scenario. When you're finished, save the game.

Kings

You've already done a lot to define the personalities of each civilization's leader. Now it's time to ordain the details of each ruler's relationship with the other civilizations in the scenario. This is unnecessary in some scenarios, but is especially vital if the political situation is a significant aspect of your scenario's atmosphere and excitement. Even if it's not, treaties, attitude, reputation, and so on can add spice to an otherwise lukewarm scenario.

All of the options you need are under the Edit King feature on the Cheat menu.

- ▼ **Edit Treaties** is perhaps the most important of the political options. This is what you use to set who has what treaty status with whom—including not just cease-fires, alliances, and peace treaties, but also wars and vendettas (hatred beyond mere war). You also control whether or not contact has been made between the two empires, and whether an embassy is in place. Note that both the vendetta and embassy are one-sided settings. To make a mutual vendetta, you must edit both kings. When using this option, keep in mind how players might take advantage of alliances and embassies.
- ▼ **Set Last Contact** enables you to determine how many turns have passed since a king last spoke with each of the other rulers in the scenario. This is useful for establishing contact between kings in your scenario who have not actually met each other. (You can also make contact using Edit Treaties.) The number of turns rarely matters.
- ▼ **Set Attitude** simply controls how each king feels about the other rulers in the scenario (regardless of whether they have made contact with one another). This is a way to set the stage for a change in the treaty status you've set up, or to predispose one civilization toward a certain demeanor with another when contact is made. Note that this only works on the AI—human players aren't bound by a predetermined attitude.

- ▼ **Set Reputation** is a way to besmirch the honor of one or more kings right from the start. This is a method of preventing trust and souring negotiations, but it only works on the AI—human players aren’t bound by other civilizations’ reputations. If you want to keep the player from being trusted by the AI, this is the route to take.
- ▼ **Clear Patience** isn’t something you’ll use when you’re creating a scenario. None of the kings has used up any patience, so there’s no reason to clear it.
- ▼ **Clear All Last Contact** is not normally necessary when you’re setting up a scenario, as none of the kings has made contact with any others. However, if you’ve made contact by accident—placed units in adjacent squares, for example—this is a handy way to undo all those mistakes at once. Just remember that this clears contact with *all* other rulers.
- ▼ **Set Research Goal** comes in handy when you want to specify the next advance a civilization will discover. It’s usually more fun for the player to have the opportunity to choose the empire’s research goal, but that might not be appropriate to your design. If there are civilizations you don’t intend the player to rule, this consideration does not apply.
- ▼ **Set Research Progress** enables you to put a civilization partway along the road to its research goal (assuming you’ve assigned one to it). This can be useful if you want the early years of a scenario to include the threat of an incipient discovery, but keep in mind that the player might be controlling the civilization in question.
- ▼ **Clear Research Goal** is something you might want to use every time you make a scenario. Rather than setting the research goal for each civilization, it’s often better (that is, it makes the scenario more fun) to let the player choose. Clearing the goal for every king makes that possible. Of course, it also gives the AI the same privilege. If there are civilizations you don’t intend the player to rule, you needn’t clear the goal for those rulers.
- ▼ **Edit Name** is normally useless if you set the names of all the tribes and rulers in *rules.txt*. If you didn’t—or if something went wrong and the names are not as you had hoped—then you can use this to give default names to the leaders and all the civilizations in the scenario.

- ▼ **Copy Another King's Tech** could have been a useful shortcut back when you were setting up the advances each civilization has, which is why I mentioned it there.
- ▼ **Toggle Female Flag** is like Edit Name—useless if you set up the rulers in *rules.txt*. If you didn't, then you can use this to decide whether each civilization's leader is male or female.

Money

Don't forget to determine with how much gold each civilization begins the scenario. This is normally a relatively minor point, but—as always—what's important depends on your scenario design.

Deciding the size of the treasury seems pretty straightforward, and it is. You open the Cheat menu, select Change Money, pick a civilization, and enter a number. What you need to watch out for are the ends of the spectrum.

Too much gold, and the civilization might be able to buy the game. Wonders, city improvements, units, enemy cities, and the friendship of enemy leaders can all be bought. Don't think that players won't figure this out and take advantage of it.

Too little gold, and the civilization could fall apart. City improvements must be paid for, and if all the trade goes to tax income, there's no science going on and no happiness generated through luxuries. Doesn't sound like fun to me.

Never give any civilization so much money that its total treasury might possibly exceed 30,000 gold. Above that, the treasury can “roll over” to a negative number (which causes other problems, as you can imagine). In a normal game, no empire ever needs anything approaching this amount. Also consider that, in a multiplayer game, one player might intentionally push another one over the limit by giving a monetary gift. Keep the treasuries reasonable, and you shouldn't experience this problem.

CAUTION

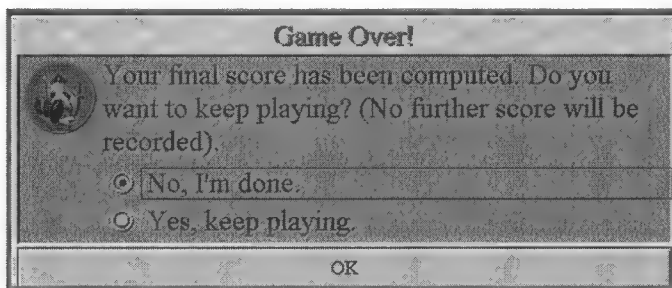
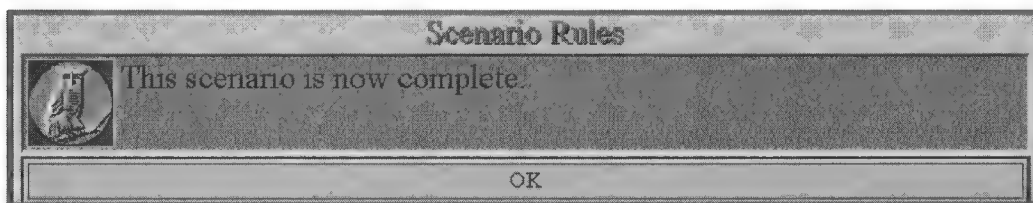


The Telling Details

At this point, if you've been following the steps as they're outlined, your scenario is essentially complete. It might not be exactly what you intend, since we haven't covered events yet, but it is a complete scenario. What's left to do—and what this little section is about—are the bits and pieces. These parameters are far from unimportant; in fact, some of them are vital to a scenario. They are, however, easy to deal with, especially since they're all in one place—the Scenario Parameters menu. (The last four are in the Special Rules submenu of that menu.)

The Passage of Time

Unless you're designing a historical scenario, you might think that the dates and timing are inconsequential. That might be true in some cases, but in most scenarios, even the most trivial details contribute to the overall impression the player comes away with. Anything that makes your game more interesting, more fun, or a more immersing experience is worth doing.



There are three options on the Scenario Parameters menu that control time in the scenario.

- ▼ **Starting Year** is almost exactly what it sounds like—the date (year or month) that the game starts. However, this isn't necessarily the year in which the *scenario* begins, but rather the first year of the *game* that the scenario is built on. The difference is the number of turns that elapse while you're setting up the situation. If you want to, you can make sure they're the same by using the Cheat menu option Set Game Year to make the number of turns elapsed zero (0). You should never, however, set the Starting Year to zero (0).
- ▼ **Turn Year Increment** enables you to specify how many years (if you enter a positive number) or months (if you enter a negative number) each game turn represents. Determining the appropriate time scale for your scenario is normally a matter of trial and error. Calculate what you think it ought to be, based on the starting date and the number of turns, then change it if it doesn't work out right (when you play the scenario). If you leave this set to zero (0), you use the default game system, which means the increment changes over time.
- ▼ **Maximum Turns** is the total number of turns the scenario can run from the Starting Year you specified. So if you don't reset the turn using Set Game Year, you eat into the scenario a bit. How long you let things go on is entirely up to you, but keep in mind that players get the option to continue after the time runs out.

How did MicroProse get a million years to pass for every turn in the Age of Reptiles scenario? You can read about it in Chapter 10: Tricks of the Trade.

TIP

Wipe/Restore All Goody Boxes

Should you leave the villages (huts, goody boxes, minor tribes, or whatever you call them) in your scenario? This parameter enables you to take them all out or leave them in, but how do you decide? Well, consider what they do:

Game Length

When you're deciding how long to let your scenario run, it might help to know how many turns there are in a normal game. It depends on the difficulty level, as follows:

Deity and Emperor = 400 turns
 King = 450 turns
 Prince = 500 turns
 Chieftain = 550 turns

- ▼ *Unleash Barbarians*—If you’ve excluded Barbarians from your scenario by limiting them to Villages Only, wiping the villages completes the job. On the other hand, you might change the text of this message and turn Barbarians into an interesting event. (In that case, you might want to change the hut icon, too.)
- ▼ *Ancient Wisdom*—This could be any advance in the scenario that the discovering civilization doesn’t have yet—including ones that you have specifically denied them. On the other hand, in a more flexible situation, a few free advances can make the game more interesting.
- ▼ *Gold*—It’s unusual that this can cause problems, but if the economic situation in your scenario is precariously balanced, any destabilizing factor could be troublesome.
- ▼ *Friendly Mercenaries*—Free units can cause problems in historical scenarios and others in which the situation is designed in detail. The mercenary units are generally just a bit more powerful than what is currently extant in the game, and the limitations of what advances the civilization has already discovered are ignored. This can be a boon to emerging empires, but a pain if you’ve explicitly limited the units in the world. Another potential problem is unique units popping out of huts.
- ▼ *Empty Village*—I don’t see how this could cause any problems, but it could be fun to change the message so that it’s more entertaining.
- ▼ *Nomads*—This is similar to the Friendly Mercenaries possibility, except that the empire gains a Settler-type unit. The potential problems are essentially the same, with the added factor of the Settler abilities—terrain improvement and establishing (or adding to) a city.
- ▼ *Advanced Tribe*—This is perhaps the most troublesome possibility, especially in precise scenarios. An instant city can throw a serious monkey wrench into your plans.

Base your decision on all of these factors. You cannot separate them out; villages do all of these things, and there’s no way to eliminate any of the possibilities. Boiled down to a single sentence, my advice is this: If your scenario design is precise, eliminate the huts; if you’re creating a flexible situation that can handle random factors, leave them in.

Reveal/Cover Whole Map

If you want every civilization to be able to see everything that's happening all over the world, this is the way to do so. (The Reveal Map option on the Cheat menu can cause more trouble than it's worth, and the Entire Map option doesn't have any permanent effect, anyway.) Also, if in the process of creating the scenario you have unmasked more of the world than you would like a civilization to see, this is your tool for re-setting their discovered area.

Before you finish with any scenario, you must decide how much of the world each empire has explored. If it's the whole world, then you've made things easy for yourself. Just use this to reveal the whole map to everyone. If, on the other hand, you intend to limit each civilization's knowledge of the map, here's how:

- 1 Cover the whole map. (You won't see the effect of this if you've used the **Cheat** menu to reveal the map, but trust me, it works.)
- 2 For each civilization, every unit on the map uncovers the area it can see. To fill in the areas in between and any other areas you want to be explored, you must create units in the areas you want uncovered, then destroy them again.
- 3 As long as you don't allow game turns to elapse, you can use units with large movement allowances (air units are best) to advantage. Create one, make it the active unit, move it in such a way as to uncover an area of the map, then destroy the unit.
- 4 Repeat this as many times as necessary to uncover as much of the map as you choose.
- 5 Switch to the next civilization and repeat the process.

This can take a lot of time, so don't be impatient. It also spends one whole game turn (one turn for each of the civilizations), so you might need to re-set the current game year and treasury, as well as the shield progress and food storage in each city.

Remember that if the Apollo Program wonder (or whatever you rename it) is possible in your scenario, when that is successfully built by any empire, it reveals the entire map to every civilization.

NOTE

Set Scenario Name

The name of your scenario is reflected in the introductory text. (Unless you remove it, that is; read Step 6: The Foundations of Civilization, for the details.) It doesn't show up in too many other places, so one could argue that it's not really important. However, leaving out little touches like this one can make your scenario seem sloppy. Give your scenario a name.

Toggle Total War Flag

The Total War flag does one thing and one thing only; it prevents the Senate from interfering in the war-making of the leaders of Democracies and Republics. This was put into the game primarily for the sake of scenarios involving historical conflicts. (It would have been pretty silly if, in the American Civil War scenario, the Union Senate forced President Lincoln to sign a peace treaty with the Confederates in the first few months of the war.)

The trouble with using this option is that it removes an important balancing element from the game. Just as a Fundamentalism without the science penalty would be too powerful, so a Republic or Democracy without the frustrations of the Senate can be too easy to rule. (It's not as unbalanced as Fundamentalism, since there's still the unhappiness issue—which for most players is the more serious limitation of these forms of government.)

Keep in mind that Total War is not necessary if the civilizations in question are prevented from negotiating with one another (via events, which are described in Step 8: Schedule Events). The Senate can interfere only during negotiations.

NOTE

The manuals for both of the scenario packs mistakenly refer to Total War scenarios as precluding spaceships. This despite the fact that the same person (me) wrote both those descriptions and the (correct) description of the Total War option in the Technical Supplement to the original *Civ II* manual. It's true that the Total War scenarios also have the Bloodlust option enabled, so that spaceships are not allowed, but the text in the manual is misleading. *Mea culpa*.

Forbid Government Switching

I've mentioned this one in a couple of other places. It's most useful in those scenarios with precise designs. For example, a scenario based on the Cold War would lose some of its verisimilitude if the Soviet Union decided to switch to a Fundamentalist government in 1960. For scenarios with open structures, however (like the Age of Reptiles scenario), this limitation is generally neither appropriate nor fun.

There are other ways to make switching governments impossible, but none is as effective. The exception is a situation in which you don't want to prevent every civilization from switching governments, just specific ones. In that case, limiting the availability of the requisite government advances is your best tactic. Giving one civilization the Statue of Liberty can be useful as well, but keep in mind that that advance might be captured—and its effect applied to another civilization than the one to which you gave it initially.

One last issue to remember is that there is no way to force a government switch once the scenario has begun—there is no “ChangeGovt” event. If you forbid government switching, it's permanent.

Forbid Tech from Conquest

This is one part of the strategy you must pursue if you intend to keep separate tech trees from intermingling. (The others are preventing negotiations and eliminating diplomatic units.) It's also a way to keep the more primitive civilizations in your scenarios from advancing more quickly (via conquest) than you intend them to.

Unless you want to slow or prevent the spread of advances, this option has no other value.

Eliminate Pollution

Depending on the situation you have set up in your scenario, pollution might become a problem. If the setup you require to create the circumstances you intend causes pollution you don't want, you have a problem. Sometimes you can alleviate this by placing pollution-reducing improvements in the cities, but that's not always an acceptable option. The fact is, there are times when you just don't want to worry about pollution in your scenario. That's what this parameter is for.

Keep in mind, however, that without the limiting threat of pollution, civilizations can build industrial infrastructures that might outrace even what you intend for the scenario, and they can toss nuclear weapons around without fear of environmental consequence.

One of the reasons this option might be attractive for your scenarios is the fact that the AI in *Civilization II* never makes any effort to clean up pollution. It's true.

TIP

Special WWII Only AI

When the game manual says not to use this one, it's not kidding. You should pretend this option doesn't exist, because there's no way you can use it productively.

The WWII scenario was designed for the most part by a programmer. Unfortunately, in early versions of the scenario, the computer-controlled civilizations did not act like their historical counterparts. To overcome this failing (and because he could), he wrote a specific set of instructions for the AI in that scenario. Those instructions cannot be applied to any other scenario, and *Civilization II* is likely to crash if you try it. ☒

STEP 8

SCHEDULE YOUR EVENTS

If you've read the entire book to this point, you know everything you need to know to create complete scenarios. Events are not necessary to make a good scenario; they're the tool you use to make good scenarios *magnificent*.

Adding events to a scenario is both exciting and frustrating. You can do really cool stuff, but it can be difficult to get it right. This chapter explains some of the do's and don'ts, provides a few examples, and explains the known quirks of every trigger and action. Sorry, but if you have only the original *Civ II* game and neither of the scenario packs, you cannot add events to your scenarios.

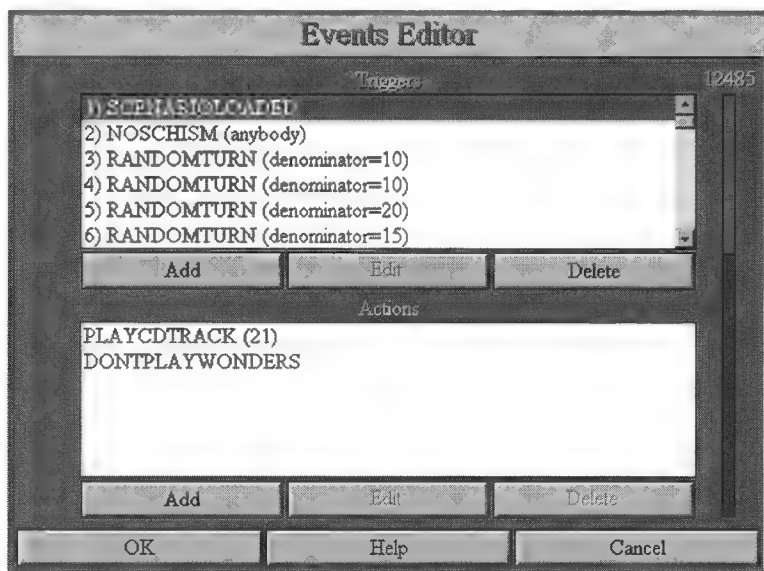
Important Notes

Before we get started, there are a few things you really should know about the process of creating an *events.txt* file. These are relatively minor quirks, but if you're not aware of them, you could run into unanticipated major problems.

Stupid Editor Tricks

If you have *Fantastic Worlds*, you have a decision to make when it comes time to schedule the events for your scenario. You can use the handy Events Editor built into that scenario pack (see Figure 8-1), or you can make and edit your *events.txt* file the old-fashioned way, with a text editing program. The catch is, you can't do both.

Figure 8-1: It's this or a text editor—not both.



If you create *events.txt* with the built-in editor, then later make changes in the file using some other method, the scenario will crash. If you create your events file without using the Events Editor, then later modify the events using the Events Editor, the scenario will crash.

Both the Events Editor and text editor methods work just fine and enable you to do everything you need to do, so there's really no reason to switch from one to the other. Still, this can be a bit of a problem if you forget.

Versions

The two scenario packs, *Conflicts in Civilization* and *Fantastic Worlds*, include two similar but different versions of the event language. For those of you who have *Fantastic Worlds*, the changes are laid out at the end of the manual. For the rest of you, here's the short list of what's new in the second scenario pack:

- ▼ The RECEIVEDTECHNOLOGY trigger was added.
- ▼ The AnyUnit wildcard parameter was removed.
- ▼ Three new wildcard parameters were added: *TriggerAttacker*, *TriggerDefender*, and *TriggerReceiver*.
- ▼ The heap dedicated to events was increased to 32 Kb.
- ▼ Three new actions were added: GIVETECHNOLOGY, DESTROYACIVILIZATION, and CHANGETERRAIN.

If you are converting a scenario created with *Fantastic Worlds* for use with *Conflicts in Civilization*, you must make sure to remove any of the added stuff for the events to function.

If you have the version of *Conflicts in Civilization* that is included in the *Civilization II Multiplayer Gold Edition*, you have the updated events parser. What does that mean? It means you can use events just as if you had *Fantastic Worlds* which you do. The new trigger, new actions, and new parameter wildcards that were added for that scenario pack are in *Civilization II Multiplayer Gold Edition*, as well as the few fixes MicroProse made. Note that this also means you cannot use the *AnyUnit* wildcard.

NOTE

What's a Heap?

The manuals for both scenario packs mention that there's a dedicated heap for events—16 KB for *Conflicts in Civilization* and 32 KB for *Fantastic Worlds*. Maybe some of you already know what this means, but your average human being thinks a heap is just an unruly pile.

In this case, “dedicated heap” is just a fancy way of saying that there's only a limited amount of working memory (RAM—random access memory) reserved for events.

All the events you schedule are loaded into memory when the scenario begins, and they're held there waiting to be activated while the game goes on. (This is an inexact explanation, and a good programmer could tell you a lot more, but it's essentially correct.)

The end result is easy to understand; you can't go hog wild with events. Every event takes up some memory, so you can put only a limited number of triggers and actions in each scenario. How do you know how much is too much? If you use the Events Editor, you can consult the memory gauge included with that utility. Otherwise, you're stuck with humankind's oldest method—trial and error. If none of your events work and you're certain that there's nothing else amiss, chances are good that you've exceeded the heap.

Creating the Events File

In contrast to all the complex, interesting, and fun things you can do with events, the process of putting them into your scenario is remarkably simple. All you do is create a text file, put some text into it (in the right format), and make sure the file is in the right folder. If you're using the Events Editor in *Fantastic Worlds*, even these few steps are automated for you.

Detailed and complete instructions for creating a properly formatted *events.txt* file are included in the manuals for both scenario packs. I'm not going to waste your time by repeating them here.

Notes and Quirks

The possibilities for events are as nearly infinite as your imagination can make them (within the limits of the memory allowed for the heap, anyway). It would be foolish of me to even try to tell you everything you can do with them. What I can do, though, is give you a few illustrative examples—and tell you a few things about each trigger and action that the manuals don't cover.

For a few more interesting samples of events—culled directly from existing scenarios—take a look at Chapter 10: Tricks of the Trade.

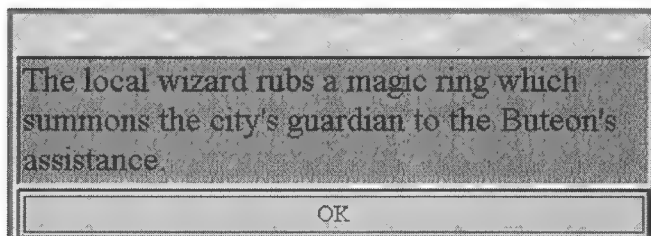
Triggers

For the most part, the event triggers are pretty straightforward. You can usually predict the function of a trigger from its name. In this section, I discuss each trigger in more detail than there is in the game manuals, hopefully without repeating too much that you already know. If there are quirks in the way a trigger functions, I clarify those, too.

One notable thing missing from the game manuals is examples of valid triggers. The chart is nice, but since it's so important to get the format right, illustrations would have been helpful. To clarify the proper format, I've included a correctly formatted sample of each trigger.

CityTaken

Like most of the triggers, CITYTAKEN is uncomplicated and potentially quite useful. When the city you specify by name is controlled by the civilization (*Defender*) you specify, then is captured—or bought—by the other civilization (*Attacker*) you specify, the actions you've included in this event are initiated.



Obviously, this trigger is great for historical scenarios. Certain cities are always valued more highly than others, and you can simulate the importance of regaining (or retaining) a city by attaching the MOVEUNIT action to this trigger—forcing defensive units (or reinforcements) to travel as quickly as they can to the conquered city. Here's an example of the valid format for this trigger:

CITYTAKEN

CITY=Prague

ATTACKER=Australians

DEFENDER=Czechs

Of course, military movements are not the only potential use for this trigger. By combining this with the DESTROYACIVILIZATION action, you can make a particular city *really* important (but you must warn the player about something with such radical consequences). You can also reward someone—not necessarily the attacker or defender—by giving out money or technology when a city is captured. The CRE-ATEUNIT action enables you to design your own version of the “Instant Partisans” popping up in the normal game.

Cities, especially those in contested areas, can trade hands frequently. If you wish to limit how often this trigger gets triggered, think twice about using the *Anybody* wildcard—be specific about one or the other (or both) of the civilizations that must be involved. If you’d rather it only happen the first time the city is breached, make sure to include the JUSTONCE action in the event.

Negotiation

This is one of the easiest triggers to accidentally misuse, but it’s also a great way to prevent civilizations from making deals that would go against your intentions for the scenario.

Essentially, this trigger does two things. First, when the civilizations you’ve specified (by name or by type) attempt to talk, it prevents the negotiation. Second, it activates any actions you’ve included in the event. This means that even if you have *no* actions in the event, this trigger still prevents the negotiations. Here’s an example of the valid format for this trigger:

NEGOTIATION

TALKER=Outcasts

TALKERTYPE=Human

LISTENER=Anybody

LISTENERTYPE=Computer

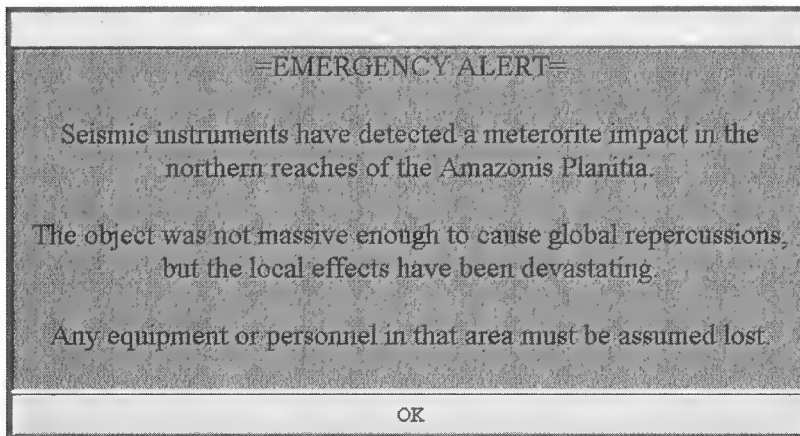
This trigger is great for preventing negotiations between civilizations, which is what it was designed for. Outside of that, I’ve found few good uses for it—which is not to say that you won’t find some that I haven’t. As the manual says, computer-controlled civilizations try to talk to each other so often that attaching actions to their attempts

results in overkill. You can use the JUSTONCE action to limit the consequences to one-time effects. However, if you do so and still want negotiations prevented, you'll need to put in a second NEGOTIATION trigger without the JUSTONCE action.

Where this trigger comes in handy is in a situation when you know exactly which civilization the human player is ruling. (This is uncommon, but in some scenarios—like X-COM Assault—there's only one civilization the player is supposed to rule.) If you schedule penalties or rewards for attempting to negotiate, a human player can learn to avoid or take advantage of the events you've created—the AI cannot.

RandomTurn

As triggers go, this is probably my personal favorite. I've always believed that *Civilization II* had too few random factors. (I don't mean that I think luck should be more important to winning than skill—that's gambling, not a strategy game—but that too much determinism can become pretty dull; a few chance events can add spice.) This trigger enables designers to add a little chaos to their scenarios.



The denominator is important here, as is a little understanding of probability. Every turn, there is a chance—one chance out of the number you specify—that the event will be triggered. Regardless of whether or not the event is triggered during a turn, the chance of it happening again the next turn is exactly the same. Unless you include the JUSTONCE action in this event, it is possible—but unlikely—that the event will be triggered *every turn*. It's also possible that it will *never* be triggered.

Here's an example of the valid format for this trigger:

RANDOMTURN

DENOMINATOR=50

You can use any of the possible actions successfully with this trigger (except for DONTPLAYWONDERS, which belongs with the SCENARIOLOADED trigger). The key is to alert the player that something has happened. I suggest that you always include a TEXT action with RANDOMTURN, announcing to the player what has happened. For a few examples, refer to Chapter 10: Tricks of the Trade.

ReceivedTechnology

This particular trigger can be problematic, but only if you're careless. On any turn that the civilization you specify *has* the advance you specify, this trigger is activated. That means on the turn that the tribe acquires the advance in the first place *and* every turn thereafter, until the end of the game.

NOTE

This trigger was introduced in *Fantastic Worlds*. If you do not have that scenario pack or the *Civilization II Multiplayer Gold Edition*, you can't use this.

This is useful for attaching actions to the discovery of an advance, like a text message warning the player that, "The Little Pigs have developed Brick Houses." In that case, however, you want to include the JUSTONCE action in the event. (No sense warning the player every turn for the rest of the game that the pigs can build brick houses.)

The possibilities for this trigger are myriad. You could start a war over a discovery, attach a monetary prize to it, create a one-time special unit (researching *Reanimation* generates Frankenstein's Monster), destroy a civilization (the discovery of *Iron Working* poisons all the Elves), and so on. Here's an example of the valid format for this trigger:

RECEIVEDTECHNOLOGY

RECEIVER=Aztecs

TECHNOLOGY=25

So, when would you *not* make this a JUSTONCE event? Well, let's say you want a certain advance to confer an ongoing financial benefit—the *Tax Reform* advance, for example, could result in a per turn income (CHANGEMONEY) for an empire—or the creation of a special (not too powerful) unit every turn. In those cases, you should keep the advance in question near the end of the game, so as to avoid overdoing it. You might also want to add a penalty—an ongoing financial loss or the intractable hatred of another ruler (MAKEAGGRESSION), to an advance that is otherwise too powerful.

ScenarioLoaded

There's not much to say about this trigger. There are only two actions you can use with it, and both of those are general switches (DONTPLAYWONDERS and PLAYCDTRACK).

The valid format for this trigger is just the word itself, on a line by itself. For the sake of completeness, here's an example:

SCENARIOLOADED

This trigger has no use other than to activate those two actions. You can't do anything else interesting with it.

Turn

Despite being extremely simple, this trigger is one of the most useful and often used of the bunch. When you want to schedule actions for a specific month or year, there's no better way. This is also the one to use if you want something to happen every turn. Clearly, this is highly useful when you're re-creating historical events—declarations of war, troop movements, financial changes, scientific discoveries, and so on.



If you use this to have an action or set of actions happen every turn, don't overdo it. For example, if you put up the same text message over and over, the player is likely

to get bored and quit. The same goes for playing a sound repeatedly—unless it’s a very short sound. You could use this to give or take a small amount of money every turn, or to make sure that two civilizations remain at war (though a better way to do that is to prevent negotiations). If you use the CHANGETERRAIN action with this, you can create a “zone of death” in which every unit and city is destroyed every turn.

The only common mistake you need to avoid is thinking of turns as equivalent to years. Each turn takes exactly the amount of time you set in the Scenario Parameters (discussed in detail in Step 7: “Scenario” Means “Situation”). Therefore, the number of the turn is *not* the same as the year—unless, of course, your scenario starts at year zero and each turn is one year. Here’s an example of the valid format for this trigger:

TURN

TURN=92

Note that events you program using this trigger never need the JUSTONCE action. By its nature, this trigger is either a one-shot deal or an every turn event you wouldn’t want to limit to one occurrence.

TurnInterval

The TURN trigger covers both ends of a spectrum; you either get a one-shot event or the “every turn” alternative—which can overwhelm the game if it’s not used carefully. Think of the TURNINTERVAL trigger as your tool for handling all of the middle ground.

This trigger is great for simulating annual events (every 12 turns if your scenario proceeds by months), legendary incidents (every 50 years, the monster emerges from its cave to ravage the countryside), and that sort of thing. It’s not much good with one-shot actions like DESTROYACIVILIZATION and GIVETECHNOLOGY.

Here’s an example of the valid format for this trigger:

TURNINTERVAL

INTERVAL=10

Simulating Seasons

It might seem that you could simulate seasonal changes using two resonant intervals and CHANGETERRAIN actions that cover the same area of the map—Tundra turning to Swamp in the summer and back again months later, for example. It's possible, but the problem is that any units or cities in the area are destroyed every time the terrain changes. If you really want to do this, read the sidebar, Simulating Seasons.

Unless you set the interval too short, this trigger is not subject to the limitations of the “every” TURN alternative—that is, not using text messages and sounds. Using the JUSTONCE action with this trigger would be silly.

UnitKilled

The UNITKILLED trigger is a great way to penalize or reward civilizations for losing their own units or destroying those of their enemies. It's also helpful for adding atmosphere in historical scenarios—text messages declaring the death of unique units or important victories, for example. You should keep a nice, somber WAV file handy for the occasion.

The following event definition causes the “changing of the seasons” discussed under the TURNINTERVAL trigger. Every sixth turn, the area in question becomes Tundra. Every twelfth turn, it then changes to Swamp. The result is that every six “months,” the area switches between Tundra and Swamp, as if it were melting in the summer and freezing in the winter.

```
@IF
    TurnInterval
    interval=6
@THEN
    ChangeTerrain
    terraintype=6
    maprect
    2,2,30,2,30,14,2,14
@ENDIF
@IF
    TurnInterval
    interval=12
@THEN
    ChangeTerrain
    terraintype=8
    maprect
    2,2,30,2,30,14,2,14
@ENDIF
```

This is great “window dressing” in a scenario, but I don't recommend using this in an area you expect to be occupied, because every unit and city in that area will be destroyed every six turns.

Alexander the Great and his elite Companion Cavalry are defeated in battle. Alexander dies of his wounds, however his generals vow to carry on. The Macedonian Greeks, are limited to a Marginal Victory.

OK

Another important use for this trigger is to cause the replacement of units you don't want destroyed—by linking it with the CREATEUNIT action. In fact, most of the actions work well with this trigger. Mick used UNITKILLED in some creative ways when scheduling the quests in the World of Jules Verne scenario (see Chapter 10: Tricks of the Trade for the whole scoop). Here's an example of the valid format for this trigger:

UNITKILLED

UNIT=Archduke

ATTACKER=Anybody

DEFENDER=Prussians

One final note on this: if you use UNITKILLED with the MAKEAGGRESSION action, don't waste your time having the civilizations involved in the killing (*Attacker* and *Defender*) declare war on each other. If a unit has attacked another unit, war has already been declared between them. Instead, use MAKEAGGRESSION to activate alliances and involve third parties in the conflict.

Actions

Like the triggers, most of the event actions are relatively straightforward—they have exactly the effects their names lead you to expect. What I do in the following sections is present each action in a little more detail than the game manuals (without repeating too much of what you already know), and outline any quirks in the way the action functions.

One thing the game manuals don't do is give examples of valid actions. Considering how important it is to get the format just right, that would have been helpful. To mitigate this oversight, I've also included a correctly formatted sample of each action.



ChangeMoney

This action is exactly as simple as it sounds. You can change the amount of gold in a civilization's treasury by adding or subtracting a specified amount. *Civilization II* prevents you from subtracting too much and forcing the treasury below zero.

Here's an example of the valid format for this action:

CHANGEMONEY

RECEIVER=Shylocks

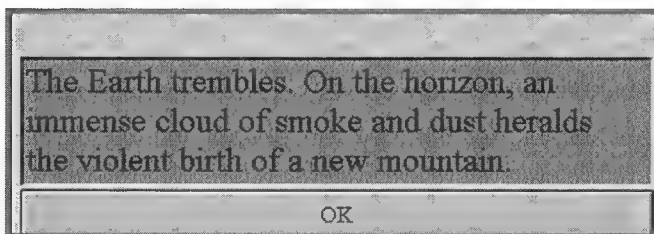
AMOUNT=100

There are no quirks in this action, but there is one potential problem. Remember how, when you were setting up the situation (in Step 7: "Scenario" Means "Situation"), I cautioned you not to give any civilization so much money that its total treasury might possibly exceed 30,000 gold? (When the treasury gets above 30,000, it can "roll over" to a negative number and ruin the whole game.) The same limit applies to this action. It shouldn't become a problem—unless you've set the cost of things far too high, no civilization will ever need that much gold.

For this reason, you should be wary of using CHANGEMONEY in conjunction with repeating triggers such as TURNINTERVAL and RECEIVEDTECHNOLOGY and with triggers that get activated often, such as NEGOTIATION. Stick to small amounts in these cases, and test-play the scenario a lot to make sure you can't exceed the 30,000 gold limit.

ChangeTerrain

This is possibly my personal favorite action. I've used it to simulate meteorite strikes, dust storms, volcanic eruptions, the growth of new mountain ranges, and a rising land bridge. This is how Mick destroyed Atlantis (among other things; see Chapter 10: Tricks of the Trade for more examples). The possibilities for this one are almost endless, but there's one fly in the ointment. If you change an area of terrain to type 10 (Ocean), all of the cities and units in that area are destroyed, but any existing city caught on the *edge* of the changed area—just offshore of the new coastline—is not completely eliminated. The city icon is gone, and for all normal game purposes—attack, production, zones of control, and such—the city does not exist. The trouble is that *Civilization II* doesn't believe it's gone; the civilization that owned that city can *never* be eliminated from the world. To some of you, that might seem like an interesting tool to play with, but unfortunately it's not fun—even if the player owns the immortal, useless city.



NOTE

This action was introduced in *Fantastic Worlds*. If you do not have this scenario pack or the *Civilization II Multiplayer Gold Edition*, you can't use this action.

Since you can never tell for certain where the civilizations in the game might build cities (unless there are no Settler-type units at all), **CHANGETERRAIN** actions involving type 10 should always be all-inclusive. That is, you either destroy an entire continent (with a margin for error), or you don't use type 10 at all.

Don't forget that when you specify the map coordinates, you must use the in-game coordinate system, not the Map Editor system (for the details see Step 2: Create Your Map). Here's an example of the valid format for this action (taken from the Age of Reptiles scenario):

CHANGETERRAIN

TERRAINTYPE=5

MAPRECT

52,92,54,92,54,94,52,94

This is a fun action to experiment with. In the MicroProse scenarios, it's used most often with timed triggers (**TURN**, **RANDOMTURN**, and **TURNINTERVAL**), but that's no reason to limit yourself the same way. Under the right circumstances, the conquest of a city (**CITYTAKEN**), the destruction of a special unit (**UNITKILLED**), or the discovery of a dangerous advance (**RECEIVEDTECHNOLOGY**) might trigger all sorts of changes to the surrounding terrain.

CreateUnit

Generating units out of thin air is one of the most useful tricks in the event designer's book. It's especially handy if you've created units that you've set aside so that no civilization can build them. (Once again, the details are in Chapter 10: Tricks of the Trade) That's one way to have monsters—or unexpected reinforcements—suddenly appear.

Yes, you can create units for the Barbarians this way, but remember that Barbarian units frequently disappear if they don't encounter a unit or city within a few turns. Also, note that unlike nearly every other reference in the game, this action requires you to specify the type of unit *by name*, not by slot number. Even the most minor spelling error in the unit type or the name of the civilization will cause the action to misfire—no unit appears, and you might cause an error, too.

Don't forget that when you specify the map coordinates, you must use the in-game coordinate system, not the Map Editor system (for the details see Step 2: Create Your Map). Be careful—you can create units outside their Domains, but you don't want to (they won't be able to move, and the game might crash). Here's an example of the valid format for this action:

```
CREATEUNIT
```

```
OWNER=Goblins
```

```
UNIT=Hobgoblin
```

```
VETERAN=no
```

```
HOMECITY=none
```

```
LOCATIONS
```

```
121,47
```

```
59,29
```

```
ENDLOCATIONS
```

One unusual use for this action that I've seen used is to "prevent" the destruction of a particular unit. For example, say you've given one empire a unique wizard unit. You could create an event that triggers on the defeat of that wizard (UNITKILLED), and creates a replacement wizard (CREATEUNIT) near a friendly city. A text message (TEXT) could explain that the wizard escaped by teleporting magically. A similar event could, if it was appropriate to your scenario, create two wizards (or more!) for every one that was killed.

DestroyACivilization

There are no fiddly details with this one. The civilization in question just vanishes. The only legacy it leaves is the record of its existence. (History is not wiped out.) Thus, for example, the Wonders that civilization built cannot be built again.

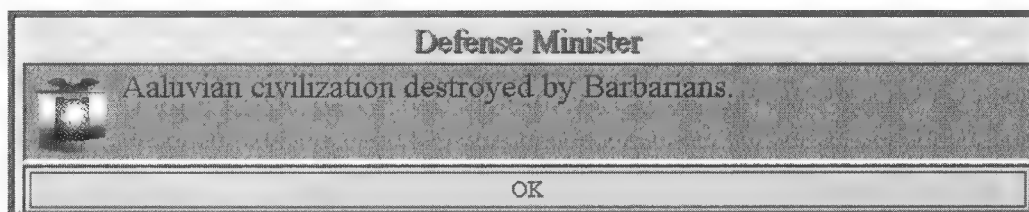
NOTE

This action was introduced in *Fantastic Worlds*. If you do not have this scenario pack or the *Civilization II Multiplayer Gold Edition*, you can't use this action.

Here's an example of the valid format for this action:

DESTROYACIVILIZATION

WHOM=Despoilers



You should be careful with this one. You never know for certain which civilization the player will choose to rule. If you just wipe that empire out for no reason, you're going to frustrate your player pretty severely. There are at least two ways to mitigate those feelings. One is to simply warn the player in the introductory text (before the choice of tribe is made) which tribes get wiped out. The other, more interesting, method is to link the destruction of the civilization to a trigger that the player can

prevent—for example, the conquest of a special city (the source of magical power) or the fate of an important unit (the queen bee). In this case, too, you must notify the player of the situation if you want anyone to play your scenario more than once.

DontPlayWonders

Actions don't get much simpler than this one. You can use this to switch off the Wonder videos, so that they don't play during your scenario. This is a good idea if you've renamed any of the Wonders of the World, because the videos are inappropriate.

This action is most often used with the SCENARIOLOADED trigger, so that it takes effect from the start of the scenario. It's not necessary to use it that way, and if you want to play some but not all of the Wonder videos, switching them off later in the scenario might fit your needs.

This action is just one word with no parameters, but for the sake of completeness, here's an example of the valid format for this action:

DONTPLAYWONDERS

Generally speaking, experienced *Civ II* players have seen the Wonder videos enough times that they've long since switched them off (using the Graphic Options on the Game menu). Thus, this action might seem unnecessary. I recommend using it, just to be sure.

GiveTechnology

You know how valuable and powerful advances can be, so you shouldn't give them away lightly. Still, if you're working out a historical situation, this action is quite handy for simulating the actual timing of particular discoveries and inventions.

This action was introduced in *Fantastic Worlds*. If you do not have this scenario pack or The *Civ II* multiplayer edition, you can't use this action.

NOTE

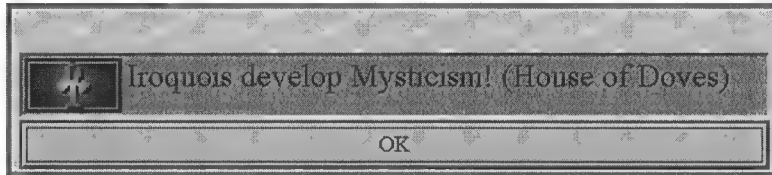
One limitation that MicroProse might undo in a future *Civilization* product is that you can't use this action to take advances away—you can't force a descent into barbarism. (Progress can't be undone; the *Civilization II Multiplayer Gold Edition* is basically an optimistic game.)

Here's an example of the valid format for this action:

GIVETECHNOLOGY

TECHNOLOGY=57

RECEIVER=Iroquois



It's not a good idea to link this action with any of the repeating triggers or those that get activated more than once. Why would you want to give the same civilization the same advance again? In fact, unless you're giving away Future Technology (number 90—the only advance that an empire can gain multiple times), this action should always be accompanied by the JUSTONCE action.

Mick used this one with great success in the World of Jules Verne scenario. Check out Chapter 10: Tricks of the Trade, for some interesting examples.

JustOnce

There's no reason to ever use this action by itself; it should always appear in an event that has at least one other action. In fact, this action isn't really an action; it's a switch.

The JUSTONCE action is the tool you use to prevent events from happening more than once. For example, all of the events that help define the quests in the World of Jules Verne scenario include this action. (The details are in Chapter 10: Tricks of the Trade.) After one civilization has completed a quest, it's done and gone. You can't discover the source of the Nile twice, nor can you be the first to overcome the legendary monster after someone else has already done so.

This is especially helpful when you're using the NEGOTIATION and RECEIVEDTECHNOLOGY triggers. You might want something to happen only the first time two civilizations meet—a text message like the one in the manual, for instance—but not afterward. The technology trigger is somewhat inconvenient,

because it is activated not only the turn the civilization gets the advance in question, but continues to trigger every turn thereafter. (It should have been named “HASTECHNOLOGY,” but no one asked me.) The JUSTONCE switch is the most convenient way to avoid problems with that trigger.

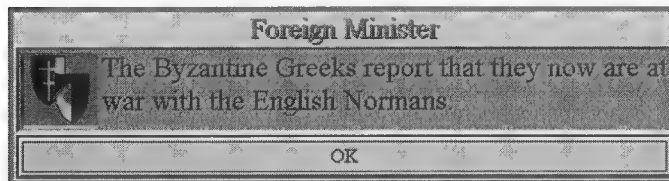
There’s really nothing to misunderstand about the format for this action, but for the sake of completeness, here’s the usual example:

JUSTONCE

The one important problem with the JUSTONCE switch involves saved games. When a scenario game is saved and reloaded, all of the events are reloaded as well. The saved game has no memory of which events have already happened, and thus JUSTONCE events can happen again. This is true in *Conflicts in Civilization*, but was fixed for *Fantastic Worlds*.

MakeAggression

The action that starts a war is most commonly used in conjunction with the triggers UNITKILLED, CITYTAKEN, TURN, and RECEIVEDTECHNOLOGY. In the first two cases, it’s used to bring a third party into an existing conflict. After all, if the Romans have just killed one of the Greeks’ units or captured a Greek city, those two civilizations must already be at war. The MAKEAGGRESSION action can force the Egyptians into the fracas. (Maybe an Egyptian ambassador was killed in the attack.)



The other two cases are more interesting. Using TURN with MAKEAGGRESSION is the preferred method for re-creating a historical outbreak of hostilities in a particular month or year. Sometimes, it’s too much trouble to arrange the situation so that war is inevitable. Think of this as a shortcut.

The RECEIVEDTECHNOLOGY trigger (available only in *Fantastic Worlds*) can be fun. (The Americans develop Fast Food, therefore the rest of the world declares war in self-defense.) The only caveat is that you must include the JUSTONCE action along with the MAKEAGGRESSION action. Otherwise, the declaration of war will be renewed every turn until the end of the game.

Note that this action is not necessary for preventing two civilizations from negotiating with one another; the NEGOTIATION trigger by itself can prevent negotiations. Adding this changes the event so that one civilization declares war on the other every time they try to meet. That's a very different thing.

Here's an example of the valid format for this action:

```
MAKEAGGRESSION
```

```
who=Exterminators
```

```
whom=Vermin
```

The names you specify for *Who* and *Whom* must exactly match the Tribe defined in *rules.txt*, not the Adjective. (Tribe and Adjective are described in Step 7: “Scenario” Means “Situation”.) You cannot use the “anybody” wildcard, either. This action is one of the reasons scenario players are not allowed to change the name of their civilization. If they could, they'd have an easy way to sidestep your intentions to have the computer-controlled tribes to declare war on the player.

MoveUnit

The computer-controlled civilizations don't know what you expect of them. To a certain extent, you can set up the scenario in such a way as to guide the AI in general directions, but once the game begins, the only instructions you can give to the AI are a few select actions. This is one of those actions.

The MOVEUNIT action combines two steps in one. First, you throw out a net (*maprect*) to catch *ground* units of a particular type and nationality. Note that point; the manual does not tell you that this action affects only ground units. Once you've caught the units, then you tell a certain number of them where to go. Which units actually receive your orders is not under your control. Ineligible units—fortified, sentried, nuclear, and those with Goto orders—are weeded out first. Next, the number of units you specified are picked out in the same order that they would normally have been activated; they may as well be chosen at random. Unfortunately, this usually means that you cannot tell precisely when those units will arrive at their destination. This puts some limits on how useful this action is in historical re-creations.

Typically, you'll use this action with triggers that might realistically be cause for troop movements—a city is taken, a unit is destroyed, or negotiations begin. In historical scenarios (or any one in which precise timing is required), you might use this to impel armies in a certain month or year.

Note that it is perfectly acceptable to set the destination map coordinates to be a city you want attacked. Don't forget that when you specify the map coordinates, you must use the in-game coordinate system, not the Map Editor system (for more details see Step 2: Create Your Map). Here's an example of the valid format for this action:

```
MOVEUNIT  
  
UNIT=anyunit  
  
OWNER=Federals  
  
MAPRECT  
  
73,33, 73,33, 73,33, 73,33  
  
MOVETO  
  
71,35  
  
NUMBERTOMOVE=all
```

This particular action is most useful at the beginning of the scenario, when you know exactly where everything is. It's also handy immediately after you create new units—again, because you know where they are. If you use this action later in the game, the only way you can be sure to catch the units you want in your *maprect* is to specify an entire continent—or even the whole map—as the collection area.

PlayCDTrack

In theory, this action enables you to decide what piece of music plays at the beginning of the scenario. You should hear that track in its entirety, except when situational music is playing—during negotiations, for example. After that piece has played, *Civilization II* chooses the next track (and all those after it) pretty much at random.

You can play any track on the CD that's in the drive, with two exceptions. Track 1 is off limits, because it is used for program data. The other limitation is that *Civilization II* can't recognize any track beyond 24. Here's an example of the valid format for this action:

```
PLAYCDTRACK  
  
2
```

You should know that assigning music to a scenario is a hit or miss affair. (A programmer I know took a look at the source code for this part of the game and remarked, “I’m surprised it ever works right.”) Most of the time it does work, and you get the music you’re after, but at times you might not. If you have trouble with it, don’t be too surprised.

PlayWaveFile

This action is as simple as it seems. It plays the WAV file you specify, and there are no complicating factors I’ve been able to find. Here’s an example of the valid format for this action:

```
PLAYWAVEFILE
```

```
    nukexplo.wav
```

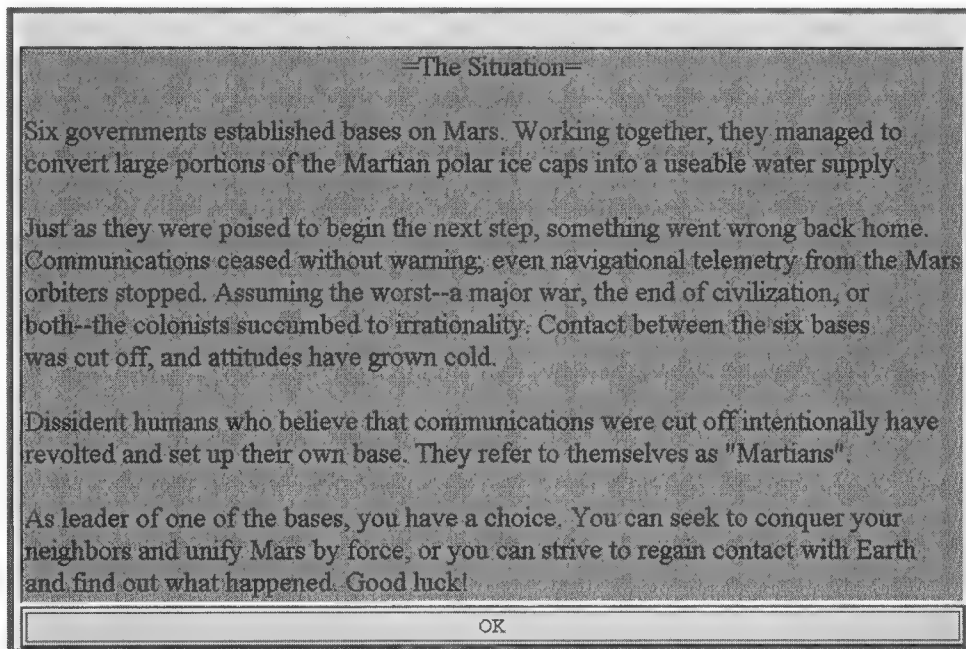
The catch is that the file you play must match the usual *Civ II* sound format. In case you’ve forgotten (or haven’t yet read Step 3: Design Units), *Civilization II* can only play a WAV file if it is in this format:

- ▼ **22kHz**—This is the only sampling rate that’s guaranteed to work. If you try to use anything else, you’re on your own.
- ▼ **8-bit**—The game works well with 16-bit sounds, but that’s *only if* you’re not using a pre-95 version of Windows *or* you have a 16-bit sound card. Versions of Windows earlier than 95 (3.1x) cannot play 16-bit sounds on an 8-bit sound card.
- ▼ **Mono**—Once again, if you have Windows 95 (or newer) or a 16-bit sound card, you can probably use stereo sound effects without a problem. Otherwise, forget it.

Like the next action, TEXT, this one comes in handy when it’s used in concert with one or more other actions. For example, if you change the amount in a civilization’s treasury *and* you want the player to know about it, you could play the *newbank.wav* file (which sounds something like a cash register).

Text

There’s really nothing quirky about this action that I know of. It’s so straightforward that it would be difficult to make it complicated. The only thing you need to avoid is putting in so much text that the box overflows the screen—or the player gets tired of reading.



The manual doesn't tell you that the text can be more than one line. There are also a couple of formatting tricks you can use that the manual doesn't mention.

- ▼ A caret (^) marks the beginning of a new line. Therefore, a line with nothing on it except a caret (^) is displayed as a blank line.
- ▼ A line of text that starts with two carets (like this: ^^) is centered in the box.

Here's an example of the valid format for this action:

```
TEXT
^^Admiral Spiffy's flagship has been sunk!
^
^The evil RapsCALLion Empire is responsible.
^The Military Advisor suggests that we smite them!
ENDTEXT
```

The TEXT action is most useful when it's used in concert with one or more other actions. This particular example would obviously be triggered by the destruction of a unit (the flagship), and the same event might include an action that causes one of Admiral Spiffy's allies—not a civilization the player is supposed to rule—to declare war on the Rapsallions. ✂

STEP 9

FINISHING TOUCHES

When you're ready to admit that your scenario is complete—all the design and planning is complete, the art and sounds are done, everything is in place, and it all seems to work—that's when it's time to wrap it up and put the bow on the box.

This step is about the wrapping. You might have noticed that the MicroProse scenarios all start out with a spiffy opening picture (it's called a *splash graphic*) and some introductory text, plus each has a specific piece of music associated with it. Here's how you can add these features to your scenarios.

Making a Splash Graphic

It's remarkably easy to assign any picture you want as the “splash graphic,” the picture that pops up first, behind the introductory text for the scenario. (See Figure 9-1 for my favorite example.) The difficult part is getting it to look right.

Figure 9-1: The splash graphic for the Age of Reptiles Scenario



Here's what you need to do:

- 1 First of all, you need to create or find a picture you like. If you're drawing one from scratch, you should make sure that the graphic conforms to the following list of formatting criteria. If you're using a graphic file that you found somewhere (legally, I assume), you must convert the file to meet the same criteria.
 - ▼ The graphic file must be in 8-bit GIF format, like all of the other *Civ II* GIF files.
 - ▼ The file must use the *Civ II* palette. Most graphics editors include a tool for importing (or saving and loading) a palette from an existing file. Use that to get the palette from one of the original *Civ II* GIF files and apply it to your graphic file.
 - ▼ The image should not be larger than 640 pixels wide and 480 pixels high (in short: 640 x 480). Anything else won't fit on the screen in *Civilization II*'s lowest resolution—and some people do play at 640 x 480 resolution.

- 2 When you have the file ready, complete, and fitted to the required criteria, place it in your scenario subfolder (not the folder named *scenario*, but the folder in which you're keeping the scenario the graphic is for).
- 3 Change the name of the file to **title.gif**.

That's all there is to it.

Writing the Intro Text

After the splash graphic sits on the screen by itself for a few seconds (while the program loads a few files), the text introduction to the scenario appears, covering up most if not all of the picture (see Figure 9-2 for an example). This text is drawn from a text file in the scenario subfolder. When you create this text file, you must give it the same filename as the scenario file (but a different extension, of course). For example, if your scenario file were named *gerbils.scn*, the introductory text file would need to be *gerbils.txt*. For convenience, I'll refer to the intro text file as *scenname.txt*.

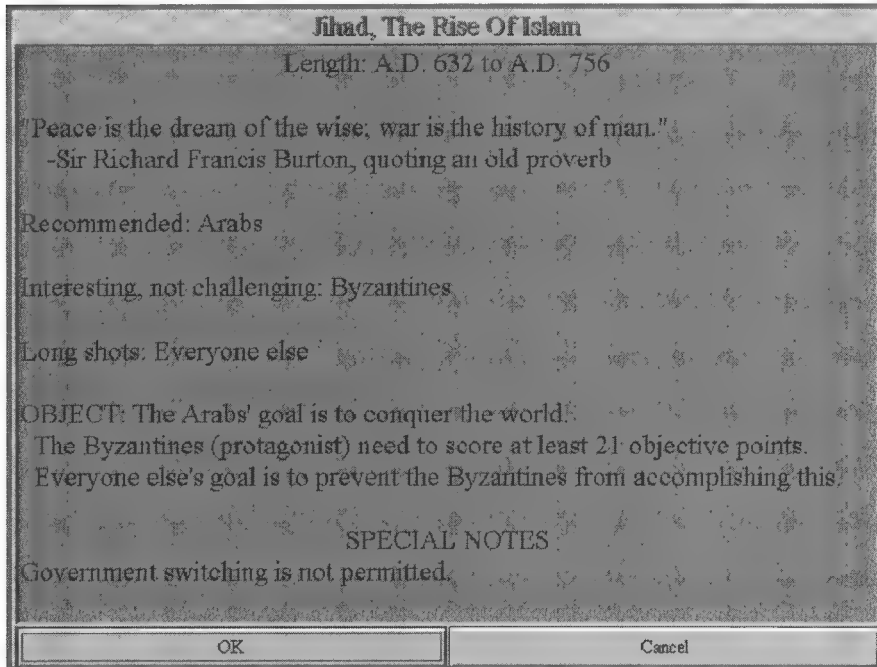


Figure 9-2: The introductory text for the Jihad scenario

The *scenname.txt* file must be in text only format, and the first line in the file *must* be:

```
@SCENARIO
```

The second line determines the width of the box in which the text is displayed. It must be:

```
@width=450
```

except that you can change the number (at the end) to best suit the text for your scenario. Notice that there are no spaces before or after the equals sign; this is true of the title line, too. Make sure you don't put any in by mistake.

The third and fourth lines are optional. If you want your introductory text to have the name of the scenario as a title, use this as your third line:

```
@title=%STRING0
```

If you prefer to give the text a specific title, you can replace the `%STRING0` part with any text you like—as long as you keep it short.

If you want the length of your scenario displayed, use this as the fourth line in the file:

```
^^Length: %STRING1 to %STRING2
```

Scenario Data Variables

What all this `%STRING` stuff really boils down to is that there are three variables you can use to insert data from your scenario into the introductory text:

- ▼ `%STRING0` represents the name of the scenario.
- ▼ `%STRING1` is the scenario's starting date.
- ▼ `%STRING2` is the end date of the scenario (the starting date plus the product of the game length and the amount of time you've set to pass per turn).

The rest of the file is where you write the introductory text for your scenario. You really don't have a lot of space to work with—just to the bottom of the screen—but you can do a lot in a little space. There are some formatting tricks you can use, too.

- ▼ Every line must begin with a caret (^).
- ▼ A line with nothing on it except a caret is displayed as a blank line.

- ▼ A line with nothing on it at all is ignored—not displayed.
- ▼ A line of text that begins with two carets (like this: ^^) is centered in the on-screen box.
- ▼ No special characters are available—just plain text.

Last, but not least, every intro text file *absolutely must* end with the line:

```
@end -- this line must be here!
```

Assigning Music

Every scenario can have a specific piece of music assigned to play at the beginning of the game—whenever the situational music isn’t playing, of course. You assign this piece of music in the *events.txt* file. You can play any track on the CD that’s in the drive, with the exception of Track 1 (which is used for program data) and any track beyond 24.

Unfortunately, because you must assign the scenario-specific music using the events file, those of you who have only the original version of the game cannot do so.

CIV II

I’ve already covered events in Step 8: Schedule Your Events (and the instructions in the manuals aren’t bad either), so I’ll assume that you understand the basics of events and won’t repeat all that here. To assign music to your scenario, insert the following as the *first* event in your *events.txt* file, replacing the # with the number of the track you want played:

```
@IF

SCENARIOLOADED

@THEN

PLAYCDTRACK

#

@ENDIF
```



CAUTION

Frankly, assigning music to a scenario is a hit or miss affair. Most of the time it works, but sometimes it doesn't—and sometimes it works on one computer, but not on another. The interaction between the portion of the game program that handles music and the other part that schedules and runs the events can be quirky and—I hate to say it—a little unreliable. If you have trouble with it, don't be too surprised.

If you've assigned a track to the scenario but no music plays when you start it up—rather than the wrong music—the problem is one you can fix. Reload your latest normal save file (not the scenario file) and open the Game menu. Pick Game Options and make sure that the Music option is turned on (checked). Now save the scenario again—both ways. When you load the new scenario file, the music should be playing.

Testing, Tweaking, and Balancing

This is the last step, but only in a manner of speaking, because it's really more than one step. Also, this step can and sometimes will involve undoing and redoing *all* of the other steps. Of course, I'm talking about testing.

When the scenario is done (you think so, anyway), you can just dump it out there—give it to your friends, post it on your web site, or whatever you intend to do with it—but you take the risk that there's something wrong with it that you don't know about. No matter how careful you are, no matter how organized, little errors creep in. Any experienced software developer will tell you (sometimes grudgingly) that there's no substitute for playtesting.

So here's what you need to do. First of all, play the scenario yourself. Play it a lot. Play it over and over and over. Play every civilization in it. Try to win every possible way. If it's still interesting and fun the twentieth time, that's a good sign. If it surprises you, and the AI actually makes you work to win, that's even better.

Next, give it to someone else to play. Choose someone who knows games pretty well and, if you can find one, pick someone who's a better *Civilization II* player than you. (Note that I said a better *player*, not a better *designer*.) If you know someone who has worked as a playtester, try to talk him or her into it. Explain to your guinea pig that you want lots of detailed feedback. What I usually say is, "Tell me what's wrong with it. Break it if you can." You might need to do this several times with a few different people, because you won't always get good testing from the first person you find. Keep at it; it's worth the trouble.



Last comes the hardest part of all. Whatever your testers say is wrong with your scenario—your brainchild that you’ve spent all this time sweating over—you must take them seriously. It’s human nature to try to rationalize away complaints, because making changes is hard—and once you’ve told yourself a job is done, it’s really (really) tempting to ignore feedback so that you don’t need to go back and redo things. Don’t fall for it. Be strong, be honest with yourself, and fix what’s wrong.

The worst problems—other than crashes, of course—tend to generate the least specific comments. If one of your testers says, “It was kind of easy,” you have a *serious* problem. Or maybe one says, “It was alright, but I had way too much money and couldn’t get my cities to grow.” Statements such as these are clues that the game balance of the scenario is off. It’s likely that your playtesters won’t notice when details are wrong (they don’t know what the new city icons are *supposed* to look like), but their comments regarding the game play should clue you in as to what is wrong with the bigger picture—which is harder to fix, but much more important than the little things.

This can turn into a long back-and-forth process. You make a new version, they test it and tell you what’s wrong, you fix things, they find more, you fix more, and so on *ad nauseam*. Eventually, you have to make a judgment call. At some point, the scenario is as good as it’s going to get, and it’s time to push it out of the nest (ship it!) and go on to the next one. ∞

PART II

The Designers' Notebook

This part goes beyond basic scenario building tips with stories, tricks, and advice from the game designers at MicroProse.

TRICKS OF THE TRADE

A big part of what makes a scenario unique is often some or another trick the designer managed to make *Civilization II* do. In this little chapter, I open the curtain on some of the magic and illusion practiced in the scenarios MicroProse has released.

With the single exception of the special AI in the WWII scenario, the designers at MicroProse haven't cheated—anything we've done in a scenario, you can do, too. Here are some of the more useful and interesting tricks we've used, dissected for your enlightenment.

Private Tech Trees

Quite a few scenarios take advantage of a trick first developed by Mick Uhl for the Alien Invasion scenario—separate tech trees (or at least branches) for different civilizations. This one is not as simple as most, and you’ve got to take some care to prevent problems, but it has proven to be useful in a wide variety of situations.

Here’s how Mick does it:

- 1 Select one advance and remove it from the scenario by replacing both its prerequisites with **no**.
- 2 Build a self-contained branch of advances by ensuring that at least one prerequisite of every advance in the branch stems from another in the branch and that they all inevitably lead back to the removed advance.
- 3 Using the **Edit Technologies** option on the **Cheat** menu, endow at least one advance from the branch—other than the removed advance—to the selected tribe or tribes. It’s best to give an advance that has the removed advance as one of its prerequisites. Note that you cannot get the same effect by giving the removed advance to a tribe, because *Civilization II* doesn’t recognize it as a prerequisite and, thus, would never present any advance in the branch for selection. No other tribe can break into the branch, because the removed advance is necessary to do research in the branch, and it is not in the scenario.
- 4 Once the detached branch is finished, test the scenario thoroughly to make sure you haven’t given any tribe an unfair advantage.

This technique is successful in preventing other tribes from *researching* their way into the private branch, but it doesn’t prevent them from *trading*, *conquering*, or *stealing* their way into it. To prevent civilizations from invading private branches of science, you must forbid technological gain from conquest, remove diplomatic units from the scenario, and use events to foil negotiations between the affected tribes.

Limited to a Marginal Victory

In some of the scenarios that involved important personages from history—Alexander the Great and Napoleon, for example—the units that represent those persons are deemed so vital that if they are lost, their civilization is limited to a Marginal Victory at best.



This trick is exactly that—a deception. As anyone who has played one of these scenarios through to the end and done well already knows, the actual score (and the type of victory) takes no notice of the destruction of the irreplaceable unit. It's just a text message—an event triggered by the destruction of the unit.

Unique Units

While we're on the subject of those important historical personages, just how do you set them up, anyway? No one can build them—they're just *there*—and they can't be replaced. Sometimes they even pop up after the scenario has begun.

Mick has used this trick in a few of his scenarios, including Alexander the Great and The Age of Napoleon. It's one of the first he invented, and it goes like this:

- 1 Pick an advance you can afford to leave out of the game and turn that one off (make the prerequisites **no** and **no**).
- 2 Pick another advance that you can leave out of the game, and make the removed advance a prerequisite for that second advance. This way, no tribe can ever research this advance, but it is still in the game.
- 3 Don't give either of these advances to any civilization.
- 4 Make the second advance the prerequisite for the unit. Since no tribe can ever get this advance, none will be able to build the unit.
- 5 Place the unit in the scenario yourself, using either the Create Unit option on the Cheat menu or an event with the *CreateUnit* action.

The first thought most designers have when they want to try this is simply to make the unit obsolete so that no one can build it. We discovered, through trial and error, that if you do that, once the tribe that controls it gets that obsolescence advance, the AI might just decide it's not worth maintaining and disband it.

You also can't just turn off the prerequisite advance—the game won't recognize it. Even if you give the advance to a tribe first and then turn it off, it'll still stay in the list, but won't work that way. If it's an obsolescence advance for a unit, it will work, but it won't function as a prerequisite advance. That's why you have to go one step beyond—use two advances—so that the unit will stay in play.

Private Units

If you understood that last one (Unique Units), then you're ready to create units that can only be built by specific tribes. The methods are related, and I'm sure a few of you have already figured it out. This particular artifice has been put to use in many different forms and in a number of scenarios.

What you need to do is to guarantee that a particular unit is available only to one particular civilization, and that no other tribe can build it. The order in which you do things is important in this case. First of all:

- 1 Pick an advance you can afford to leave out of the game, but don't change its prerequisites yet.
- 2 Assign that advance as the obsolescence advance for the unit or units you want to limit access to.
- 3 When you're setting up the scenario, give that advance to all the tribes you don't want building the unit.
- 4 Save the scenario.
- 5 Now, go into *rules.txt* and set the prerequisites for the advance to **no** and **no**, thus removing it from the game.

The difficulty we ran into was keeping the tribes that you *do* want to build the unit from ever getting the obsolescence advance. There are plenty of ways to do that, but this one is much more graceful and powerful than the others—it takes advantage of a quirk in the game program. Once you've assigned the obsolescence advance to the tribes you don't want to build the unit, you can turn that advance off, and it will still work to make the unit obsolete. Once turned off, it's not available for research, trade, capture, or theft—it's completely out of play except for the obsolescence effect.

If you want every tribe to have its own set of units that other tribes can't build, you use the same method, but expand it a bit:

- 1 First, determine how many tribes there are going to be (3 to 7), then pick out that number of advances that you're willing to remove from the game. (Don't forget that you can use the extra advances for this.)
- 2 In your mind, link each of these advances to one particular tribe. You might want to write down which is which, because you'll need to refer back to it later.

- 3 When you're setting up the scenario, give the advance you've linked to each tribe to all the other tribes, but *not* to the tribe it's linked with. That is, you give each tribe all the advances you linked to other tribes.
- 4 Save the scenario.
- 5 Go into *rules.txt* and turn off all those linked advances. (Set their prerequisites to **no** and **no**.)
- 6 Now, all you need to do to make a unit available to only one tribe is to look at your list of advances, then make the advance you've linked to that tribe the obsolescence advance for that unit.

The Holy Lance and True Cross

Those of you who have played the Crusades scenario will recognize this one. If you're playing as one of the crusading factions, you gain the ability to build two elite units—the Knights Hospitaller and the Knights Templar—only after you have gained hidden advances. The Holy Lance allows the Templars, and the True Cross makes the Hospitalers possible. Both of these advances can be traded and stolen, but neither is available through research. The only way to get them is through the capture of a city—an Islamic city. Not only that, but even though the Islamic nations already have these advances (else how could you capture them?), they do not and cannot build the elite knights units. What gives?

This one is simpler than it appears. The trick is to play games with advances.

- 1 Decide what advance will allow a tribe (we'll call them the "Christians") to build the special unit, and also what advance will make that unit obsolete.
- 2 Put the definition of the unit into *rules.txt*.
- 3 When you're setting up the situation, you prevent other tribes (the "Moslems") from ever building that unit by giving them the special unit's obsolescence advance.
- 4 Next, give the Moslems the advance that makes the special unit possible. It's of no use to them, so, as Mick says, "it just sits there."

Now, if the Christians get the advance that makes the unit possible from the Moslems—by capturing a city, for example—they can build the unit. Of course, there’s never any guarantee what advance anyone will get when they capture a city, but that’s not your problem. Sooner or later, the right advance will come out.

To make sure that the Christians were forced to capture a city to “find” the advances, Mick used the NEGOTIATION trigger to prevent negotiation between the civilizations involved.

How do you make certain that the Christians never gain the obsolescence advances for the elite units? (After all, you don’t want all those special knights to just up and disappear.) After you’ve taken all of the above steps (and saved the scenario), go back into *rules.txt* and take the obsolescence advance out of the game—change its prerequisites to `no` and `no`.

Remember, you can give an advance to a tribe while you’re designing the situation, then later remove that advance from the game. The obsolescence effect of the advance will continue to function, but in every other way, that advance is out of play.

If you have only the original version of *Civilization II*, there’s still a problem. The Moslems still possess the obsolescence advance, and the Christians could capture it when they take a city. This was fixed in *Conflicts in Civilization* and all subsequent releases.

500 Million Years BC

If you’ve been paying attention, you probably noticed that the Age of Reptiles scenario begins some time in the Triassic era (500 million BC) and runs until the Cretaceous-Tertiary boundary (65 million BC). There’s not enough room in the Starting Year box for numbers in the millions, and *Civilization II* doesn’t recognize any date earlier than 4000 BC, so what’s up?

Well, this is a case of a minor text change producing a nice effect. If you take a look into the *labels.txt* file for that scenario, you’ll see that the second line under the @LABELS header has been changed. What is normally “B.C.” now reads “Million B.C.” instead.

Sinking Atlantis

In the game, the sinking of the islands of Atlantis is a monumental cataclysm, but in design terms, it's nothing more than a single event. Mick combined a random turn trigger with a text message, a terrain change, and a nice sound to get the effect. Here's the text of the event that does it:

```
@IF

RANDOMTURN

denominator=401

@THEN

JUSTONCE

TEXT

The great volcano Thera explodes destroying the
two islands of Atlantis.

ENDTEXT

CHANGETERRAIN

terraintype=10

maprect

26,18,60,18,60,54,26,54

PLAYWAVEFILE

volcano.wav

@ENDIF
```

The key is that Mick did not forget to embellish the terrain change. Without the text message and the sound of a volcano erupting, this event would go unnoticed until the next turn, when suddenly the ruler of Atlantis would notice that something was missing.

We used the same *ChangeTerrain* action, linked to various triggers, to raise the land bridge and new mountain ranges in the Age of Reptiles scenario, to produce the dust storms and meteorite impact in the Mars Now! scenario, and many others.

Melting the Martian Icecaps

In the Mars Now! scenario, it's possible to change virtually useless Icecap terrain into life-giving Watershed (Ocean) simply by mining the ice. This is another case of how understanding the entries in *rules.txt* can point out simple ways to create nice effects.

Take a look at the entry for the Icecap terrain type:

```
Icecap,1,2,0,1,3,n,0,0,0,0,0ce,0,20,1,n,0,; Gla
```

Notice that the abbreviation for Ocean (Watershed) is in the position for the results of mining (Mine). Whenever a terrain square of the Icecap type is mined by a unit, it is changed into the Watershed type. That's all there is to it.

Alternate Ways of Winning

In three of the scenarios in *Fantastic Worlds*, there's a new way to win the game. Taking over the world still works, and sending a ship to Alpha Centauri would work if it were possible in those scenarios, but in the World of Jules Verne, Mars Now!, and Midgard scenarios, discovery alone can win the game for you.

As anyone who has played and won one of these scenarios knows, gaining the final advance that wins the game—From the Earth to the Moon, Return to Earth, or the Rainbow Bridge—doesn't actually trigger the end of the game. The discovery by anyone of one of these advances is merely the trigger for a TEXT action. The designers mean for you to have won at that point, but, unfortunately, *Civilization II* does not currently allow scenario designers to create alternate victories.

Invisible Tree Guards

The Elves' capital city is guarded by a group of Tree Guards. These guardians' woodland skills make them completely invisible in the forest surrounding the capital. Woodland skills? Nah.

The Tree Guards are invisible because they use the same icon (in *units.gif*) as is used for one orientation of the Forest terrain type (in *terrain2.gif*). Thus, they look just like normal Forest. The only tough part of this was moving the shield pixels so that the unit's icon completely covered the shield and the health bar. Of course, the Tree Guards could not move, or they might desert their posts—and become visible in the process.

The Martian Invasion

In the World of Jules Verne scenario, one of the more exciting—and potentially annoying—events is the invasion of the Martian War Machines. With virtually no warning, a whole bunch of immensely powerful units descends on the Earth and starts rampaging around. They don't capture cities, but they easily devastate any and all defenses set against them. Several turns after they arrive, all of the invaders keel over and die just as suddenly as they appeared.

Mick programmed one event to announce the impending invasion and to set it up. He used a random turn trigger to post a text message, included the JUSTONCE action to forestall any repeat invasions, and then gave an advance to one of his utility tribes:

```
@IF
```

```
RANDOMTURN
```

```
denominator=250
```

```
@THEN
```

```
JUSTONCE
```

```
TEXT
```

```
^^WORLD NEWS FLASH
```

```
Astronomers around the world notice a sudden  
and dramatic increase in activity on the surface  
of the red planet, Mars. They can only describe  
it as a sporadic series of large explosions.
```

```
ENDTEXT
```

```
GIVETECHNOLOGY
```

```
technology=11
```

```
receiver=Exotics
```

```
@ENDIF
```

Why give the advance? That's the tricky part. Mick wanted two text messages, with a one-turn pause between them. You can't put two actions of the same type (TEXT, in this case) into the same event, so Mick set it up so that the first event triggered another one. The one turn delay is a welcome side effect of the way the RECEIVEDTECHNOLOGY trigger works. The second event reads as follows:

```
@IF
```

```
RECEIVEDTECHNOLOGY
```

```
technology=11
```

```
receiver=Exotics
```

```
@THEN
```

```
JUSTONCE
```

```
TEXT
```

```
The large explosions on Mars turn out to be  
the launchings of an invasion fleet intent  
upon the subjugation of Earth.
```

```
ENDTEXT
```

```

CREATEUNIT

unit=Martian War Machine

owner=Exotics

veteran=false

homecity=none

locations

74,26

endlocations

@ENDIF

```

This event created one war machine, and the events that followed—identical to this one but without the text message—placed several more.

The Martian War Machine is an air unit, so it can never occupy a city. This was important to prevent the Exotics (the utility tribe) from accidentally becoming a world power. More importantly, though, it kept the Martians from having a city to land in. The Martian War Machines have a range of seven, which means that, as air units, they will crash after seven turns unless they land. They are intentionally created in locations that are absolutely certain to be far enough from any Exotic city that there is no way at all for any of them to ever land. If a war machine could land, it would not expire, and part of the plan is for them to die off—just like they did in H.G. Wells’s book.

Gnolam Income Bonus

In *Master of Orion II*, the Gnolam species are expert interstellar traders. The system of species’ characteristics in that game defines their financial advantages, but how did Ken Burd translate that into *Civilization II*?

The Gnomes gain 10 Gold every turn, over and above whatever they actually earn. This bonus is a single event in *events.txt*.

```
@IF  
  
TURN  
  
turn=-1  
  
@THEN  
  
CHANGEMONEY  
  
receiver=Gnomes  
  
amount=10  
  
@ENDIF
```

What's that `turn=-1` bit? That's an alternative way of writing `Turn=Every`. At one time during development, the *Every* parameter wasn't working too well, so Ken (who is also quite a good programmer) checked into the source code. He found and used this alternative method.

Entrance to Underworld

The Midgard scenario is filled with interesting stuff. Some of the best gimmicks, I've noticed, can be surprising in their simplicity. The Entrances to the Underworld (the name is shortened in the game so as to fit) are a case in point.

These portals are scattered around, and every so often, nasty creatures come popping out to sow destruction. The entrances themselves are actually special terrain resources. Have you noticed that they only appear on Underground terrain? That's why. Rather than use up one of the icons for constructions (airbase, fortress, and so on), Mick took the easy route. Not only did he save an icon for some other use, he also avoided having to place the portals himself—it's done by the map generator.

After Mick had created the scenario map, he simply noted the locations of all the portals to the underworld. Later, when he wrote the series of events that create the nasty critters, he made sure to use those locations as the places to place the monsters. The events themselves are merely a random turn trigger matched with a unit creation action.

Quests

Perhaps the most interesting—certainly the most inventive—parts of the scenarios Mick designed for *Fantastic Worlds* are the quests he built. Of course, I don't have room to go through all of them, but I can walk through a good example. The rest, you can figure out by reading through the *events.txt* files for the various scenarios.

Let's take the Mysterious Plateau first. It's a good example of one that uses several simple parts to make a complex whole:

```
@IF
```

```
UNITKILLED
```

```
Unit=Mysterious Plateau
```

```
Attacker=anybody
```

```
Defender=Aborigines
```

Mick created a tribe-specific unit called “Mysterious Plateau” for the Aborigines tribe (a utility tribe, not designed to be played). He designed the unit to look like terrain and gave it a movement of zero so that it wouldn't wander around. He then placed one unit of that type on the map. Whenever a unit of another civilization stumbled on it, that unit would either be destroyed or it would defeat the Mysterious Plateau. In the latter case, the quest event gets triggered.

```
@THEN
```

```
CREATEUNIT
```

```
owner=Aborigines
```

```
unit=Tyrannosaurus Rex
```

```
veteran=false
```

```
homecity=none
```

```
locations
```

```
35,71
```

```
34,71
```

```
endlocations
```

```
@ENDIF
```

The invasion of the Mysterious Plateau releases the Tyrannosaurus Rex from its secluded home. The dinosaur is another Aborigines tribe-specific unit. Notice that Mick gave two possible locations to create the monster. Both are very close to the location of the original Mysterious Plateau unit. The second is there just in case there is already a unit in the first location. It's important that the dinosaur be created, because otherwise the rest of the quest can never be triggered. Mick could have been more cautious and put in more locations—in case a civilization came after the Mysterious Plateau in force—but it seems to work okay.

Now there's a Tyrannosaurus wandering around. Naturally, the civilization that let it loose is going to try to kill it. (Yes, you could bribe it and romp around the map with it, but that would invalidate the rest of the quest.)

```
@IF
```

```
UNITKILLED
```

```
Unit=Tyrannosaurus Rex
```

```
Attacker=anybody
```

```
Defender=Aborigines
```

If and when someone succeeds in killing the great lizard, that triggers the next event in the quest. That's basically what quests are—a series of interesting events that work together. Notice that this event is triggered only if the Tyrannosaurus Rex is still owned by the Aborigines when it is defeated. That's what makes it unprofitable to bribe the beast.

@THEN

JUSTONCE

TEXT

^^World News Flash

An expedition has recently returned from the wilds with a fantastic story. They claim to have visited a lost world of dinosaurs. This evening they will report their findings to the Royal Geographic Society. Surprises are likely in store as several witnesses claim to have seen a giant egg in their possession! They have sold their story to the Times for 1000 gold.

ENDTEXT

GIVETECHNOLOGY

technology=92

receiver=TRIGGERATTACKER

CHANGEMONEY

receiver=TRIGGERATTACKER

amount=1000

@ENDIF

This is the payoff. Whoever kills the Tyrannosaurus Rex gets their name in the paper (the text message that every good designer puts in most events), a cash reward, and an otherwise unavailable advance. Notice how the text message reinforces the atmosphere of the scenario.

Some of the other quests are simpler than this, and some are nearly twice as complex, but all of them use the events to their utmost. Like any good designer, Mick takes advantage of any way he can think of to twist the tools at hand to the job he wants them to do.

Alternate Scenarios

When you read the next chapter, The Designers Talk, you'll find out about the scenarios that have randomly chosen, multiple starting situations. Those of you who have played the scenarios in *Fantastic Worlds* already know what I'm talking about. Some scenarios have a built-in factor that keeps the player from knowing for certain where everything is. If you have *Fantastic Worlds*, you can use this in your scenarios, too.

The procedure is pretty simple, but this is a powerful tool. You can only change certain things—the map and the situation—but those things let you do quite a lot. Here's the method:

- 1 Create your scenario as usual—all the way through.
- 2 To make an alternate start for that scenario, create a second scenario using all the components of the first. You can use a different map and set up a different situation, but remember that the units, terrain types, and everything else will be exactly the same.
- 3 When you save the alternate scenario, put it into the same folder as the first one. (Make sure to give it a different name, so that you don't overwrite the first one.)
- 4 Now, go into the scenario folder and rename the alternate scenario file. Make the extension on that file *.alt* instead of the usual *.scn*.

That's it, and you can make as many alternates as you want. (I'm sure there's a limit on the number, but no one will admit to knowing what it is.) Note that none of the *.alt* files show up in the list when a player goes to select a scenario file to play. When the player picks the first scenario file (the *.scn* file), *Civilization II* randomly selects either the selected scenario or one of the alternates. ∞

THE DESIGNERS TALK

As a writer, when I hear other writers talking about writing, I often come away fired up and inspired—motivated to write. This chapter is about causing that same sort of excitement for anyone interested in building scenarios for *Civilization II*. I've managed to get Mick Uhl and Ken Burd to share some stories about the process of creating the MicroProse scenarios for the original *Civilization II*, *Conflicts in Civilization*, and *Fantastic Worlds*.

With a few exceptions—X-COM Assault, Master of Orion, Master of Magic, Ice Planet, Mars Now!, and The Age of Reptiles—all of the MicroProse scenarios in both of the scenario packs were designed by Mick Uhl. (He was an important part of the design team for the original *Civilization II*, too.) Thus, his voice dominates this chapter. Any mistakes, however, are probably mine.

Mick Uhl

Mick has been responsible for more published *Civ II* scenarios than anyone, including all of the MicroProse scenarios in *Conflicts in Civilization*, and he loves to talk about both games and history. I sat down with him in his office to talk scenarios, and this section is the result. He starts off talking about his work on one of the two scenarios included in the original release of *Civilization II*, the Rome scenario.

Rome

I started on the Rome scenario in the final two weeks of development on the original game. Maybe I picked Rome because there are a lot of different potential nationalities in that time and that area. That was my first attempt at a scenario, and one of the things I wanted to do was to give each nationality a unique orientation or emphasis. For example, I wanted Egypt to be a little ahead in technology—because of Ptolemy and their emphasis on science. Rome would have the best military units, and Carthage would have a good navy. Other Greek states, the Selucid and Antigonin Greeks, would have larger empires, so they got bigger portions of the map—and then the Celts would be in there as sort of a random factor. I was surprised during the research to find out that the Celts had actually gotten into central Greece just before the time of Alexander and had even formed a little nation in central Anatolia—they had gone into central Turkey—so they had a much larger impact than I expected. That’s how it ended up: three Greek states, Rome, Carthage, and the Celts.

I set it up so that the scenario started with Pyrrhus’s invasion of southern Italy. That was the first time the Romans had seen war elephants. I pretty much didn’t bother with changing the city improvements or research or anything. I set a time limit and used the city objectives. The object was for Rome to try to conquer as much of the world as possible.

As I said, that was my first attempt at a scenario—I didn’t change much. The big thing I tried was to do a reasonably accurate historical setup and give each of the tribes some special advantage. I could have added new units, but then I would have needed new artwork. Considering that I had only two weeks, it didn’t seem like a good idea to make more work necessary.

I had fun with the Celts, because I managed to get them into Turkey and central Greece and Spain. I always think of the Celts as being in France and up in Germany, and in fact the Celts a hundred years after this scenario begins sacked Rome, and the Romans hated the Celts. Usually, the Romans would incorporate any tribe they conquered into their empire and make them citizens. The Celts, however, they just wanted to destroy.



Conflicts in Civilization

For each scenario that I did, I always tried to do something new and different, to make the approach a little different, so that it didn't feel like you were playing the same scenario over and over with different names for the tribes. I wanted to make each scenario have some feature in it that none of the others had.

Alexander the Great

I actually finished the Alexander the Great scenario for the original *Civ II* game but it didn't get put in. The map went in. We had a plan that we would put a new scenario on the MicroProse web site each month. Alexander the Great was going to be the first one put up on-line, and we actually did that—we put it up for people to download. This was before the idea of doing a scenario pack came up.

The idea is that you're Alexander the Great. You're off in one corner of the world, and the object is to conquer as much of the world as possible—but you have this time limit. It's a race against time. So I had the Egyptians in there, and I split off the Phoenicians, though I think they were part of the Persian empire at the time. I made them independent so that Alexander wouldn't have to go directly into Persia—he would have to deal with the Phoenicians separately—and of course there was the famous historical battle against Tyre, when Alexander had to build that ramp across the channel, because Tyre was actually on an island off the coast. It was a major engineering marvel of the time. The ironic part of it was that he couldn't get into the city via this rampart, but he got the Phoenicians to focus so much attention on the building of this ramp that he managed to do a little sneak attack on another side and still took the city.

Again, I tried to give each tribe something to make them different from the others. The Indians, who were way off to the east, I gave them elephants. The Persians, their units weren't quite as good as Alexander's, but they had more of them, and they had a larger empire. Alexander started off with a very small nation, but as he captured cities, his position improved. Just as in the Roman scenario, the Celts played a large role. Alexander had to decide whether to slow his march down in order to take the couple of Celtic cities in the Baltics or ignore them and try to charge across the straits and into Asia Minor as quickly as possible. He also had to decide whether to split his armies down into Egypt or head into the heart of the Persian empire. He eventually wanted to get to India. India started small, but I was hoping that, given the time I allowed before they had to run into any of the other tribes, they could build up their forces. So, by the time Alexander got there, they might have built up a pretty formidable empire.

The problem with that game was that I didn't get a lot of feedback before I finished it. QA (the testing department) was heavily dedicated to some other project, and they weren't able to give this one much attention. Again, I didn't have any artwork changes. At the time, most of the work was building up the map—putting the road networks and cities in, trying to get the right mix of units, and all. I was still—well, I *am* still learning, still making mistakes.

We did add more artwork for the disk version, the one that went into *Conflicts*. We put special figures in for Alexander and the Immortal Cavalry. I didn't really change the game, though.

World War: 1979

World War '79—in terms of the actual design—didn't take too long. I spent the most time building up the world, and that took a tremendous amount of time. That was one of the longest, if not *the* longest, scenarios to build up, because I had to put all the cities in and populate them. We were starting in 1979, there were 6 or 7 major powers—all with full-fledged armies, navies, and all—and I had to put in all the man-made terrain improvements, all the irrigation, farmland, and all that. That was very time consuming. In terms of actually designing, I don't think I did anything very innovative for that one.

The idea was to get you into a nuclear war and into dealing with nuclear pollution—all that while managing a modern empire and fighting a modern war with all of the modern units. Generally—and I think it's this way for most people—I get just so far into the game and I either decide that I've won, because I can see that I will, or I don't want to do the bookkeeping. The modern game is time consuming—I like it, I enjoy it, but I don't get into it that often, at least in *Civ II*. I did it a lot more often in the original *Civ*. So, this was an opportunity—I was thinking of myself in this regard—to set up a game in which you can just start off in the modern game and see how all the special units work and interact with one another. I could play with the Alpine Troops and Paratrooper and all that.

So, that was the purpose of World War '79. It was a different subject—a modern warfare—and it got the player into using the best units. There wasn't much research or anything else. The war automatically happens. It's not an event; we hadn't even considered the events macro yet. This was one of the games I was working on when the proposal came up to do the scenario pack, *Conflicts in Civilization*. It was originally to be the second on-line game, but then we decided to save it and put it in the pack instead.

The Age of Discovery

By the time we started the scenario pack officially, I was almost done with the Age of Reconnaissance—The Age of Discovery. There's a very famous book that I had in college called *The Age of Reconnaissance*. That covered that whole period, so I had to change the name of the scenario to the Age of Discovery.

I tried to be a little more clever in this one. Instead of letting everybody see the entire world, I tried to give each tribe a different view. So the Spanish—Columbus was about to discover the New World—the Spanish had a view much deeper into the Atlantic than the rest. The rest of Europe just had an idea about Europe; they didn't really have much of an idea what was going on in Africa and the rest of the world. One of the tribes just represented the American Indian tribes.

The big problem I had with that scenario was choosing a protagonist nation, because I didn't want this to be a conquer-the-world scenario. If it's not a conquer-the-world, then you have to use the territorial objectives—the city objectives system—and that requires a protagonist. Who would the protagonist be? All the European powers are fairly equal, and if you made the Spanish the protagonists, they'd be easily overwhelmed unless you made them ahistorically strong. Then, if you played the Spanish, the game would be too easy. What I did was an experiment. I kind of combined—I'm not sure that it worked properly—all the third world powers of the time—Moscow, China, India—into one super tribe. Then I tried to restrict them so that they couldn't grow, but were just there as a nuisance. I was at a very experimental stage, trying different things.

I still didn't add any new units or new advances into that scenario. I was trying to make an interesting situation. It was hard, because that was a full world, and it took a lot of time to populate, get the terrain right, and get all the units in. I also tried to make all the major European cities pretty invulnerable—tough to attack. So, if a nation was to succeed, they had to spread out and go into the new world, go into Africa, and go into Asia. They were forced to try to build their empires that way, rather than focusing on Europe, which was just a small part of the game. That was the first time I actually had to go in and clear off different parts of the map (by placing units) for each tribe so that they all saw some different piece of the world. At the time I thought, "That's pretty cool."

One thing I learned from the process of that scenario was that in the final two weeks, you shouldn't be designing anything new. You should just be testing and tweaking and making sure everything works.

Jihad: The Rise of Islam

Jihad was the first scenario in which I made serious use of the events macro, and this was when I ran into my first problem with the victory conditions. I wanted to use the territorial objectives, where basically the cities are points and each civilization tries to capture them. The trouble was, I couldn't determine who the protagonist should be. I wanted, of course, the protagonist to be the Arabs, but they started off in a very weak state. They were one of the weakest of the starting powers, and my fear was that if they were the protagonist and you were playing as one of the other tribes, it would be too easy to wipe them out and win right away. So, I ended up making the Byzantines the protagonists. It was a compromise of necessity. It wasn't my first choice, but it seemed to work out pretty well.

I used the events macro to prevent a lot of negotiation. Also, I said to myself, "With the Arabs, let's try Fundamentalism." I don't know how many people use it in the regular game, but I rarely do. This was an opportunity to set up a tribe with a Fundamentalist government and special, fun units.

Jihad has, I think, an interesting goal. You start off in a very small position. You're the weakest of all the factions on the board, but you need to conquer the world—or as much of it as you can possibly get control of. I think it was an interesting objective. That was the idea of Jihad, to take a weak Arab state in a small portion of Arabia and then, through the fervor of the new religion, see how far you can spread against more powerful countries.

The Age of Napoleon

Games that start in later periods generally give more trouble, because you have to do a lot more work setting them up. You begin with a lot more forces, which means that the gaps between turns—as the computer processes—are longer. The non-player tribes have large armies, so you as designer have less control over what's going on, because the more units are under AI control, the less you can predict what they're going to do. With only a few units, you can see what the AI plans to do, but with a large number of units, well, then things can kind of go backwards.

You're trying to get these tribes set up so that they'll go in the direction you want them to go and do what you want them to do—at least in the beginning—and you can't insist on, for example, "The French have to go to Moscow." Still, you want the French to be aggressive, and you try to build that into the tribe. You make them less friendly to the other tribes, make their reputation bad, and do all the things you can to get the war to continue, because that's what the scenario is about.



This was the first scenario in which I actually included units that represented individual people—units that could not be built. I had a Wellington unit, I had a Nelson unit, and I had a Napoleon unit. How did I do all that? Look it up in the previous chapter, *Tricks of the Trade*.

I had real concerns about the Napoleon scenario. Generally, I'm most worried about making sure that the start is historical. Units should be where you would expect them to be if you know the period. For this one, I wanted to start when Napoleon was in Egypt, because it's an exotic location and it allowed me to focus the player's attention—to use an area of the map which, if I'd started at a later period, nobody would really bother with. So you've got something going on down in another part of the board.

The early start also allowed the British and the French navies to do something. By a few years later, the French navies were pretty much, well, after Trafalgar they were just stuck in port. Even though that's not a big issue in playing the game, I just like to have the start at least appear historical. So then the big problem I had was letting the British fleet under Nelson get to the French fleet without the French, who move first, anticipating the British. Now, even though I made the British ships Veteran and the French not, when you attack in *Civ II*, the attacker tends to have an advantage because of the attack/defense ratios of the ships. In test games, the French were wiping out the British, and I played some games with that to try to get it to work out a little more evenly. I'm concerned with having at least the first two or three turns of the game proceed along historical lines. After that, it's up to the player—but at least I get the thing moving in the right direction.

American Civil War

The American Civil War was, on one hand, one of my favorite games to design; I thought I did a lot of interesting things with it. On the other hand, it's the scenario I'm most dissatisfied with.

My unhappiness with it stems from trying to balance it. Basically, there are two competitors in it, and they're about equal in strength. Generally, if you're playing a game with two sides, you're the human player, and if you're in a far weaker position then there's this big challenge to defeat the superior opponent. That works alright as a single-player game, but in this case, I couldn't do that. The Confederates at the start of the war could fairly be said to have been on a par with the Union side. Even though, eventually, the Union overwhelmed the Confederacy, the initial positions were fairly balanced. I tried very hard to figure out a way to keep the start historical, but make it so

that the player felt challenged—whether he played the North or the South, he should feel that it wasn't too easy.

What I did was to put both of the tribes in Democracy, and I set up the cities so that there was a likely cause of unhappiness. Both sides really had to start doing some micromanaging of the cities to get them to work at an optimum pace while also maintaining the Democratic institutions *and* staying at war—because, of course, once they went out of Democracy, production would plummet. I tried to put some extra challenges in for both players. Even so, I think that whichever side you choose, that scenario's not too tough to win. It's better if you're one among seven and you can concentrate on one tribe at a time. That's more challenging than if there's just one opponent.

There are some things I liked about the Civil War scenario. That was the first scenario in which I actually used units in roles other than units. I mean, I used a unit to actually be a place—a fort. I also built the Mississippi river—it's really just a long ocean. That was to allow naval movement up and down the river. That was also the first scenario in which we seriously put in a lot of new art; that was the first real attempt to change the artwork. We put in the occasional text newspaper headlines, about some world event while the war was going on. That was Kerry Wilkinson's idea. In a way, it's still one of my favorite scenarios. I thought the historical setup, the starting setup, was very good—it was close to the real positions at the beginning of the war—and that the units actually behaved like they should. That was a good start.

I tried to build up a trade network, so that the South could try to get blockade runners into other ports and get trade routes going. That's where I suddenly had to fight the AI. It doesn't normally recognize some of the things that go into a scenario, and it doesn't do what you want it to do. That's frustrating, but at least in the beginning of this game it appears that the ships are doing what they're supposed to do. The blockade runners try to move out of the harbors and get to some of the big ports that I represent as those little islands—the Europeans and the Mexicans and so on.

I had to put this big, huge, defensive unit in each of those island cities—it couldn't move and it couldn't be built. Those units were there just to prevent anybody from actually trying to conquer the islands. These little civs, these utility tribes, are useful. In the historical scenarios, I was using them to present the real, historical positions, plus I just wanted to have more than two countries. For example, I wanted these small, little countries in there so that the South would have someone they could trade with, and the North would have someone with whom they could set up trade routes.

The English were also in for another reason. There was a time, early in 1862, when they considered getting into the war, because of that Trent affair—the North taking



two of their Confederate representatives or envoys off of a British ship—and the British were close to going to war about that. So, this is the history you play with, and if you do something, there is the chance that the British and the Europeans could make some kind of impact. Of course, it wouldn't have been a very big impact.

I also put American Indians in, because actually, they're usually neglected, but they had a role in the western theater of the Civil War. The North was trying to keep the Oklahoma Indian tribes—the western tribes—out of the war. The South, of course, promised them more independence than they got, and so most of them sided with the South. That was an omen of the war. In the West, where there were small forces, the Indians—well, they really didn't have an impact on the scenario. They could if they attracted your attention. Still, it just makes for more interesting history. When you build a scenario, you're telling a better historical story if you include details. The Kentuckians are another example. They were neutral at the start of the war. They considered trying to stay out of the war and maintaining neutrality, so I made them a seventh tribe. If you could take their capital and conquer them, they went over to your side. The situation in Missouri was pretty much settled by that time; the South had pretty much lost Missouri. Kentucky, though, was kind of up in the air.

The War for Independence

The War for Independence was another one that gave me some trouble. It was for the same reason as the American Civil War scenario—that there are really only two major powers. It wasn't quite as bad, because the British had the preponderance of strength. They had the greater forces, so the goal of the British ruler is to get those forces onto the American continent and established in coastal cities. Then, they should be able to move in and capture the cities further inland. The Americans had the challenge of trying to build up their forces before they lost too many cities and too much territory.

Once we had the events macro language working, I was able to get the British to do pretty much what they did at the beginning, which was to abandon Charleston. They had an invasion force outside Charleston, and they made a minor attempt at it, but then they abandoned it and launched a major attack on New York City. On the first turn, the British usually capture New York, which is what I wanted them to do. That makes it more of a challenge for the Americans. Right away, they've lost their biggest city—although they've still got Philadelphia, and that's a good-sized city.

I also enjoyed putting the Indian tribes in there, because they had an impact. Out in the West, there was an effort by the colonies to try to keep them out of the war. They tried to keep the wilderness settlements in turmoil, out of the war, and away from the American settlements.

I also had the French. The problem with the French was that they really didn't get involved until they thought that the Americans had a strong chance of winning. However, England and France were at the time major rivals in Europe. When the French decided that it was worth their effort, they went in and basically won the war. I don't think the Americans—well, they might have eventually won the war, but they certainly wouldn't have won it by 1781 without the French help, because the French were able to help the Continental Army blockade Yorktown.

You can tell by the way I talk that I'm interested in the history. That was the idea for these scenarios; this is supposed to be history. When you started something in *Conflicts in Civilization*, you looked at an accurate historical position—at the start, anyway. The units tried to do—at least for the first few turns—what they historically did, and then it was up to you to play. So I had details like Benedict Arnold building his little fleet at Lake Champlain, waiting for the British invasion down there. I was more pleased with that game in terms of play balance than I was with the Civil War, but I thought they both did a good job of presenting the historical situation. Of course, with historical scenarios, you and the player can't do a lot with city improvements and advances. You find that the game is running just a very short period; it's really a conquer-the-world, warfare-focused game.

Alien Invasion

Alien Invasion was the first of the scenarios in which I was able to get out of the historical limitations and try a lot of new ideas. I had some different things that I'd been wanting to try—to see how they worked in a scenario. I guess I wanted to start pushing the boundaries of the game.

The guiding idea was based on H.G. Wells's *War of the Worlds*. Aliens come in with these super-weapons, and it's up to the human civilizations to try to work together and destroy the invaders before they're all overwhelmed. The key, of course, is to discover and build a secret weapon. There are a whole lot of things going on in there. For one, you don't know where the aliens are; they have their own city, and its location is secret. You're limited in what you can see at the beginning, and you're immediately under attack by the aliens' spaceships and warships. The idea is to try to hold them off, keep from being conquered, and research as fast as you can—up to the super-weapon that will turn the tide and make it possible to defeat the aliens. You can get to parity and be able to defeat the aliens, but it's tough. You don't know where the capital city is, and alien units are dropping all over the place and moving around, so the attacks are coming from a lot of different directions and areas. You can't just backtrack to the capital.



That was the first scenario that I put separate tech trees into. That took a long time. I had to learn a lot about how the technologies worked and how the AI treated research. Initially, I thought I'd just cut off a branch of the tree—remove those advances. Then once I'd turned off the advance at the base of that branch—given it “No, No” prerequisites—I could build the separate branches up from there. What I didn't realize was that, although advances that you turn off still work as obsolescence advances, they don't work as prerequisites. For a while I was really fighting with it, because I wanted to make it as difficult as possible for the humans to research everything needed to win—I didn't want it to be too easy. So I was reluctant to pull a lot of advances out of the human tree. I tried to squeeze as many advances as I could into the alien tech tree without making it too easy for the humans to get to their wonder weapon. It didn't help that I had to remove some advances completely and use more as base advances—connected to the removed ones—to build the trees from. (If you haven't already, read *Tricks of the Trade* for the details on how to make separate tech trees.)

The second problem was to make sure that the humans couldn't ever get one of those advances in the alien tree—they couldn't trade for it, they couldn't negotiate, and they couldn't get it through conquest. They had to be kept completely away from any of those advances. That's difficult to do, and Kerry had to do a lot of work to get the Negotiation trigger to work to stop that. That's part of the reason there are no diplomatic units in the scenario; that prevented stealing. Since there wasn't going to be any diplomacy between the aliens and the humans, that was the final thing we needed.

I thought the force fields were a pretty good idea. The theory was to keep a unit in a city and keep it sort of hidden, so the player wouldn't know it was in there. Then, when the city was attacked, the unit suddenly would appear—just like a force field. Mike Bazzell did that drawing; he thought that it was a cool idea. I had him make it big, so that it would wrap around the city. Actually, during testing, somebody in QA found out that the program would sometimes override the zero movement I set, and it would suddenly allow the force field a movement of one. Once again, Kerry had to go in and make sure that didn't happen any more.

I thought that that was a good premise for a scenario. Naming the aliens, I used the names of people that I knew. I either made anagrams out of their names, or made some joke or another. It was one way I tried to give credit to people who helped on the project.

The Crusades

With the Crusades scenario, we're once again back into history. I wanted it to start with the People's Crusade—it's also called the Peasants' Crusade—so if you look at the map in the beginning, you'll see that the Peasants are already into Asia Minor. The first thing that happens is the Turks come in and wipe them out. Sometimes some survive, because of the random factors in combat, but in many cases they all get wiped out, which was the historical case.

Then the real crusaders, the Normans and the English and the French, get going. In fact, the Normans were from Sicily, because they had emigrated. They were originally Vikings who landed in Normandy, settled there, and became Frank-icized. Then some of them moved around into the Mediterranean and got into Sicily. The problem I had with the regular crusaders was that I wanted to have them not *in* Asia Minor at the start of the game but *traveling* there. Historically, they were hit by warrior bands and so forth in central Europe, and they also had to deal with the Byzantines before they could get through into Asia Minor. That was a big element of the first crusade—the crusaders' dealings with the Byzantines—and I had difficulty trying to channel them through so that they could go through the Bosphorus and into Turkey or Asia Minor without them ending up going to war with the Byzantines. I would say I was maybe 75 percent successful. Historically, the crusaders went through Constantinople, but of course that's a Byzantine city. Even if two civilizations are allied, they can't go through each other's cities, so I moved the units through the upper straight—the Dardanelles. They went through this other one, and they started conquering cities and that worked out okay.

This scenario was similar to Alien Invasion and Jihad, in that the player's starting off with small, elite forces, and the idea is—within the time frame of the game—to conquer enough cities quickly enough so that you can eventually build up a force and defeat the other countries. So, in this game, you were going to be ruling one of the crusading factions. You didn't want to be the Byzantines; they were a little bit too strong. If you chose to rule them, you'd have a pretty easy time winning. The Turks and the Arabs, well, you could be the Arabs. They were interesting, because the situation was a lot more complex than some people think. It wasn't just Christians against Moslems. I mean, the Turks were coming in as invaders at the same time. Beginning about the time of the first crusade, the Arabs were as much concerned with the Turks as they were with the crusaders. Plus that Byzantine empire was still there, and they were also a force. Being Eastern Orthodox, they weren't particularly friendly with the Christians—up to the point of recapturing Jerusalem, yes, but they were still on opposite sides of that schism. Even the Christians would squabble among themselves. I think having a lot of elements makes that the kind of period that's good for historical scenarios. You've got six or seven different civilizations, and they all have their own little agendas and goals and so forth. That works very well.



Now the Holy Lance and True Cross, that was an experiment. It turned out alright, I think. These were special advances that I set aside—that was from my reading of the history. The Christians were looking for the Holy Cross; they thought it really existed. Then somebody also claimed to find the Holy Lance, and they believed in that, and these had a big impact on their morale. People back then believed in these type of things. The scenario idea, then, was to figure out a way that I could have, for example, the Byzantines control these advances without actually using the benefits of them—the special crusader units. So, only the Christian forces could benefit by the advances if they discovered them. You can play tricks to do that. So, if the crusaders can go in and take a Byzantine city, then they might get an advance and build the Hospitallers elite unit. That was a little innovation that I tried.

Again, I tried to restrict the negotiation between the Islamic side and the Christian side, so that the only way you could really get the advances is by taking a city, since that's what really happened. If I remember correctly, the Holy Lance was found in one of the first cities that that crusader army captured. There was some talk that somebody just invented it and claimed that he found it buried—that he was just using the “discovery” as psychological propaganda for them.

I'm not absolutely certain that it always happens the way I designed it to, because you can never get enough testing on a game. When we started *Fantastic Worlds*, that was one of my first priorities—to make sure we got lots and lots of QA time.

The Great War

World War I was sort of a tough scenario, in that, like World War '79, I wanted it to begin well along in the game. The turns run fairly long right from the start, which I think is a detriment—but if you want to play World War I, there's a lot of fighting going on, and you have to wade through that. There are some interesting units. I wanted to put the Zeppelin in. They were really the first major bombing unit in the war, and they were very effective until the allies were able to develop an engine for their fighter planes that allowed them to climb above the maximum altitude of the zeppelin. That pretty much ended zeppelin warfare, but still, it's a curious unit, so I put it in.

It took a lot of work to get the Germans to go after Paris—to go through Belgium and toward Paris. I also had to work to get the Russians and the Austrians to do what they did in the beginning. Those were the main components of the war at the start, so that worked out fairly well.

It was hard, because even though there are seven tribes, I needed to keep them in two camps, the Allies and the Central Powers. However, historically Italy switched sides,

so I didn't want to prevent a tribe from switching from one side to the other. I set it up so that the Allies were all allied with one another and the Central Powers were all allied with one another, but as the game progressed, the sides could switch around. I especially wanted the Americans to stay in the Allies. I did *not* want them to join the Germans, but that's part of the fun—the player gets to change history.

When I say I didn't want them to change over, I mean that I didn't want them to do so at the beginning—to start off doing it. I was still experimenting—it was still trial and error—and I couldn't guarantee that the countries would do what I wanted them to do. So, after it was made, I'd play maybe four or five turns into the game, and as long as they were moving along pretty much in the way I wanted them to, I was happy. After that, if they started doing strange things, well, that's the way the game is. The players are writing history—they're playing history—and we're not trying to force them into the same channels.

All these kind of scenarios are tough, because there's a lot of work you have to do on the map—a lot of cities to build and populate, a lot of terrain to improve, a lot of roads and railroads to build—but they also allowed me to put in some interesting units. You had the WWI vintage tanks in the game, and I put in Gas, which is sort of the WWI equivalent to a missile, and along those lines the super-bomber, the biplane fighters, and all these kind of strange units that were around at the time.

The Turks had to be in there. That was also the first chance to use the new battle-ships that the British had developed about ten years before—which had started the arms race and, in a way, led to WWI. I put the dreadnought type ships in. That's really it—it's the color of the period—and *that* is what you want to do in a scenario. You want to make the player feel like they're actually in that period, and that it's not something abstract. What you're seeing is what actually happened—what went on. I feel that those kind of scenarios work, in that you're challenged right from the start. You've got large enemy forces near your area (unless you're the Americans), and the idea is to try to survive, don't get overwhelmed, and build up your production base so you can start capturing cities of your own, expanding, and doing well.

In this game, too, there are some interesting things to research and some special weapons. The Germans had special infiltrating units that they actually, historically, set up in 1918 to break the trench stalemate. They were very successful, except that by then the Americans had finally gotten into the war, and it was too little too late. They got back to near Paris, but several million Americans getting into the war was just too much.

In historical scenarios, my main concern was always to at least get the starting position historically accurate, so that when you play the game you're seeing the actual situation.

For at least a turn or two, the units are moving the way they really did—which was generally the best way to move them—and the player can continue from there and become immersed in the history. Also, hopefully, every country faces a challenge. The Germans are in the center, so they have to worry about all their fronts. The French have a narrower front to defend, but their forces are generally not as strong, plus the Germans get to move first, so they have a chance to penetrate into France, perhaps take Belgium and some cities in northern France as well.

The Mongol Horde

In the Mongol scenario, again I'm re-using the idea of having the major tribe of the game in a very weak starting position, but with elite forces. It's a race against time to plan a campaign that allows you to conquer the world—or enough cities to win a victory—within the time constraint. Ideally, there's no time to dawdle; the player has to be efficient.

The Chinese are a very strong opponent. They have larger forces, even though their troops are not as good as those of the Mongols. Against the well-defended Chinese cities, the Mongols didn't have engineers, so they couldn't besiege. Historically, what they did was capture Chinese engineers and force them to do the siege. Then, the idea is to move west and go into the Middle East and, perhaps, as far into Europe as the actual Mongols did before the game ends. The scenario was really designed to be played from the Mongol point of view. That was my concern, whether it was a challenge for the Mongols. If the player wanted to play the Japanese or some other country, that's fine, but I couldn't guarantee how balanced that would be. I think the Chinese would probably have a fairly easy time.

I made all those cities visible at the start because the Mongols need to know where to go. I didn't want them to spend the game just discovering the world. I mean, they're starting the game in a campaign of conquest. Right from the start, that's what they're doing. First concentrating on the minor countries between them and China, then into China, then some of them branch off and head west—so they need to know where the objectives are. Also, historically, there were rumors. The Mongols were aware of western cities; they were aware that there was a Constantinople and some other major cities. They just weren't sure precisely where they were. I went in and opened up those areas, so that they have a general idea where the cities are, but they don't know the world in between. To do that, I just created Mongol units next to the cities and then disbanded them. It was the same method I used for the alien scenario, in which I had to give every civilization its own world view.

I enjoyed giving the Chinese a special army. I didn't put in individual leaders like I did in some scenarios, because the period of the Mongols spanned several centuries,

and there were several major leaders. If I put in Genghis Khan, then people ask why I didn't put in Kublai Khan—and they weren't contemporaries. I guess what I could've done was use the events macro to stagger the leaders, but that would also have limited the number of other units I could use, and it wouldn't have added as much to the scenario.

After the Apocalypse

This scenario I did very quickly, but by the time I got to Apocalypse, I had learned a lot by doing the other scenarios. I felt that this was a chance for me to just try a whole lot of different things. So I set aside a bunch of advances and gave each tribe its own set of units. I also tried to build in a nuclear winter; there was enough pollution that, if the player couldn't get the technology to the point of seriously cleaning it up, the world would actually change—to simulate the consequences of the nuclear holocaust.

This scenario included a lot of new units, special units. I tried to make sure that the units for each tribe reflected the personality of the tribe—the tribes had different personalities. Although, that's also the case in historical scenarios; the Mongols definitely had a different personality than the Chinese. This being a fictional account, though, I could have a little bit better control over that. I liked the fictional aspect, because it was more closely linked to the original game—the player gets almost a full, 500-turn game and pretty much starts from scratch. Each tribe had its own small set of advances that it had been able to hold on to after the war, and they could try to trade them and so on.

In fictional games, you can do a lot more than in historical scenarios, where the emphasis is generally on combat. In this one, there was also emphasis on spreading your civilization—building cities as well as conquering them. I made a much more extended research tree. The player had a whole lot more to do. Again, at the end there were the special weapons. If you could research to that point, they were very important in being able to win. Mike Bazzell did a good job. He was the main artist on the units, and he had fun with it.

I tended to enjoy this one. Even though I liked designing the American Civil War, because I like the subject, and I thought some of the ideas I had in there were good ones, I thought it was inherently a tough scenario to balance. The ones like the Alien scenario and the Apocalypse scenario were a lot more fun. I had a lot more creative freedom, and I could allow the game to play more like it was intended to play—where there is a balance of combat, peaceful scientific research, development, and all of that stuff. In fact, those two sort of laid the cornerstone for *Fantastic Worlds*. Even though I didn't plan to do another batch of scenarios, I knew once I did those that that was the direction to go if I were to do any more scenarios. Not that I don't like

doing historical scenarios, but it's hard to find a historical period that people are interested in, that has enough different civilizations to fill up the roster, and that allows you to still stay within the constraints of the original game. The Thirty Years War is definitely an ideal scenario for a war. You have countries changing sides all the time, and you have six or seven major protagonists. With that, the problem is that nobody knows about the Thirty Years War, and, at least in America, fewer really care. If I were to do another historical scenario, that would probably be the next one—and the American Civil War would be one I'd want to rework.

Fantastic Worlds

Hopefully they're all fun, but that's something only testing and playing the game can determine. They're certainly different, which is what we wanted. We want to vary the experience. People can always play the original game if they like that best, but if they want to try new things, this changes the experience. It keeps the game fresh.

The New World

I started doing this scenario, actually, before we began making the programming changes. So I was somewhat restricted, in that I wasn't sure which of the new features that were planned would actually get done and be useable. Eventually, I was able to go back and make some changes to implement some of the features I had assumed wouldn't actually get done in time.

One of the first things that I tried to do in this scenario was to make one of the tribes—well, not a non-participant, but more of a casual visitor. I meant for that civilization to have a restricted life; at a certain time I would just wipe it out, to be replaced by another tribe. I was trying to reproduce the history of casual contact that the rest of the world had with pre-Columbian North America. So I started off with the Africans, who represented a combination of peoples. I was thinking of the Phoenicians and the Romans, but I called them Africans, because you *could* lump in the Carthaginians, and it's certainly possible that some African explorers might have gotten across the Atlantic. The special thing about them was that they had two advances that the American tribes couldn't get. In fact, those advances were critical to the advance of technology in the game. To get Gunpowder and Horsemanship, you needed to contact these visitors while they still enjoyed their tenuous hold on the map—before they disappeared—and try to trade or steal or whatever to get those technologies from them. That would give you an advantage.

In this scenario, I also threw in some other interesting historical “what-ifs”. The mammoths—and some other extinct mammals—were actually present in North America when the first migrations of Asians came through. In fact, a lot of the traditional stories

go all the way back to nine or ten thousand BC, and sort of refer to hunting these huge animals. So I decided that the mammoths hadn't gone extinct, but instead the tribes were able to domesticate them, as the Indians have domesticated the Indian elephant in Asia. I basically said, "What if they had gone a different path, got this new advance called Domestication of Animals, were able to domesticate the mammoth, and used it in battles?"

Another interesting idea that popped up was the balloons. I discovered in a book that there's actually evidence for this. At one of the archeological sites—I guess in Peru—they've discovered pictures of what looked like hot air balloons. Plus, they also found a very tightly woven material that certainly could have been used as the silk for the balloon. So the speculation is that the ancient natives, at least for a short period of time, were able to make hot air balloons and fly around in them. I extended that as another possible direction to research. If you get a specific advance that allows you to learn that technology, you can use hot air balloons.

Really, there's no major European intrusion in the game. It just allows the native tribes to develop on their own, in total isolation. The tech tree starts out different than usual, but it eventually becomes normal—sort of. The assumption I made there is just that people will eventually go down the same paths to do what they need to do.

The only events that I set up dealt with the seventh tribe—the casual tribe. I pretty much got rid of it, so you started off with the Africans, but as their replacement you might end up with the Vikings or you might get the Celts. Part of the Celtic tradition is that a saint in the 1200s ventured to North America, came back, and told these tremendous stories about visiting this new land. Of course, they elaborated on them and the stories came to describe these utopian places. For native Irish or whoever else—the Vikings—probably listened to this and thought, "Oh, that's so much better than the terrible life I'm living."

Samurai

The second scenario that I completed for *Fantastic Worlds* was the Samurai. In that one I tried a few new things. Unlike the others, this wasn't strictly a fantasy scenario. It was a historical subject, but in a way it's very exotic and strange to most Americans. The medieval feudalistic period in Japan roughly parallels European feudalism. The Japanese eventually broke out of it around the first decades of the seventeenth century, when one of the shoguns finally reunited all of Japan and isolated it from the rest of the world. Then Japan pretty much remained at peace and was a stable society until the Americans came in in the 1850s.

This scenario has a lot of new and different units. It starts with the Mongols threatening to invade southern Japan. It's the early 1300s, and there's a big military force in southern Korea poised to move over to Japan. The player can rule one of four feudal clans and try to gain control of Japan. By conquering everybody else, you can eventually reunite Japan and thereby win the game. In a way, I've tried to make it historically accurate. You could bump into the Koreans, who initially have a big force—which is basically the Mongol invasion army. At that time, Korea was under control of the Mongols, and the Japanese have to consider those armies. Later on, the Japanese contrived to conquer Korea; this was part of the reunification effort. They did make two or three invasions of Korea. In fact, that's when the famous “turtle ships” that a Korean admiral had built pretty much destroyed Japanese hopes of conquering Korea by breaking up their shipping—and thereby isolating their armies on the peninsula.

There is a small European presence. They're critical in the same way that the Europeans were in the New World scenario, in that they have advances that are unavailable in Japan—specifically, Gunpowder. So if the Japanese do want to discover Gunpowder, they have to get it from the Europeans. That parallels history, in that the Spanish and Portuguese were the first to arrive in Japan. A ship accidentally ran aground in southern Japan, and the Japanese discovered the gunpowder and the rifles and the cannon on the ship. The local warlord was able to discover how to make gunpowder and copy the rifles, and like wildfire, guns spread through Japan. They became the dominant weapon, and the reunification of Japan was really mainly possible because some of the warlords discovered the potential of these guns. They built armies equipped with these guns and quickly defeated all the other warlords, who weren't quite as quick to grasp the significance of gunpowder.

The ironic thing is that as soon as Japan was reunited, the leader slowly began to ban gunpowder weapons, and 250 years later, they were basically gone. They were not part of the military system, because they were deemed too dangerous. It also had to do with the fact that you didn't need much skill to use them—they weren't the province of the samurai. It was just the general soldier that used these weapons, and the samurai class didn't like that. Some would say that Japan actually kind of went backwards in that respect.

I enjoyed making this scenario. I got to put in all sorts of samurai units and other special units—the ninja and all. I connected all the major islands, because—it's funny—naval ships were not a big part of Japanese warfare. So, units are able to cross over. I didn't want the Japanese to have to build ships to do it. Historically, they did build some, but the straits are narrow enough that communication between these

islands was fairly easy to accomplish. You didn't need to build special ships to go back and forth. Ships are basically needed if anyone wants to go to Korea or if they want to transport military units more quickly.

There was one other thing that I did right at the very end, because we discovered a problem in playtesting. Every tribe has to be present at the start of the game. This scenario starts in 1301, and, of course, the Europeans didn't make an appearance in Japan until 150 or 200 years or so after that. However, I had to put the Europeans in the game at the start. What I tried to do was hide an island off in the corner of the map—hopefully out of range of the Japanese ships initially, but not so far that they couldn't get there eventually. Trouble is, as soon as somebody plays the scenario once and discovers the European presence—and their importance—they immediately head down there the next time. So what I did was to create two maps and change the location of the European starting point. Playing one of the Japanese warlords, you're not really sure where the Europeans start; they could be in a couple of places. That made it a little more difficult to build that into a strategy early in the game. This is the first scenario in which we applied the alternate scenario files. (To learn how to do that yourself, refer to the previous chapter, *Tricks of the Trade*.)

Atlantis

The major point in the Atlantis scenario was that I actually wanted the island to disappear, just as it did in the story of Atlantis. So we built a special action in the events macro language that allowed you to destroy terrain, and I set up a random trigger that made the islands completely disappear. I didn't want anybody to know about it, because of course the Atlanteans wouldn't have known their island was going to disappear.

So you blithely go along and develop the islands—at least the first time you play—and then suddenly—it might not happen within the span of the game, but there's a good chance—sometime during the game, the islands suddenly disappear and are destroyed. The Atlanteans need to get off the two islands and onto the mainland quickly to ensure their survival. The whole plot of the scenario, the background, is based on the Greek story by Plato. Historians now believe he was hearing stories of the destruction of the Minoan civilization on Crete by the Thera volcano in about 1500 BC. Although the island of Crete wasn't actually destroyed as an immediate consequence of the volcano, the eruption so disrupted the Minoans' trade and set them back that they were vulnerable to invasion. Mainland Greeks came in within a hundred or two hundred years, captured the island, and destroyed the Minoan civilization—which at the time had been very important; the Minoans were the main Mediterranean trading power.



LICENSE AGREEMENT AND LIMITED WARRANTY

PLEASE READ THIS LICENSE CAREFULLY BEFORE USING THE SOFTWARE. THIS DOCUMENT IS AN AGREEMENT BETWEEN YOU AND THE WIZARDWORKS GROUP INC (THE "COMPANY"). THE COMPANY IS WILLING TO LICENSE THE ENCLOSED SOFTWARE TO YOU ONLY ON THE CONDITION THAT YOU ACCEPT ALL THE TERMS CONTAINED IN THIS AGREEMENT. BY USING THE SOFTWARE YOU ARE AGREEING TO BE BOUND BY THE TERMS OF THIS LICENSE. IF YOU DO NOT AGREE TO THE TERMS OF THIS LICENSE, PROMPTLY RETURN THE UNUSED SOFTWARE (INCLUDING ALL PACKAGING AND YOUR ORIGINAL, DATED SALES RECEIPT) WITHIN 10 DAYS OF PURCHASE TO THE WIZARDWORKS GROUP INC, 2300 BERKSHIRE LANE, PLYMOUTH, MN 55441 AND YOUR MONEY WILL BE REFUNDED.

1. Ownership And License. This is a license agreement and NOT an agreement for sale. The software contained in this package (the "Software") is the property of the Company and/or its Licensors. You own the disk/CD on which the Software is recorded, but the Company and/or its Licensors retain title to the Software and related documentation. Your rights to use the Software are specified in this Agreement, and the Company and/or its Licensors retain all rights not expressly granted to you in this Agreement.

2. Permitted Uses. You are granted the following rights to the Software:

- (a) **Right to Install and Use.** You may install and use the Software on a single computer. If you wish to use the Software on more than one computer, please contact the Company for information concerning an upgraded license allowing use of the Software with additional computers.
- (b) **Right to Copy.** You may make and maintain one copy of the Software for backup and archival purposes, provided that the original and each copy of the Software are kept in your possession.

3. Prohibited Uses. The following uses of the Software are prohibited. If you wish to use the Software in a manner prohibited below, please contact the Company at the address, phone, or fax numbers listed above for information regarding a "Special Use License". Otherwise, you may NOT:

- (a) Make or distribute copies of the Software or documentation, or any portion thereof, except as expressly provided in this Agreement.
- (b) Use any backup or archival copy of the Software (or allow someone else to use such copy) for any purpose other than to replace the original copy in the event it is destroyed or becomes defective;
- (c) Alter, decompile, or disassemble the Software, create derivative works based upon the Software, or make any attempt to bypass, unlock or disable any protective or initialization system on the Software;
- (d) Rent, lease, sub-license, time-share, or transfer the Software or documentation, or your rights under this Agreement.
- (e) Remove or obscure any copyright or trademark notice(s) on the Software or documentation;
- (f) Upload or transmit the Software, or any portion thereof, to any electronic bulletin board, network, or other type of multi-use computer system regardless of purpose;
- (g) Include the Software in any commercial products intended for manufacture, distribution, or sale; or
- (h) Include the Software in any product containing immoral, scandalous, controversial, derogatory, obscene, or offensive works.

4. Termination This license is effective upon the first use, installation, loading or copying of the Software. You may terminate this Agreement at any time by destruction and disposal of the Software and all related documentation. This license will terminate automatically without notice from the Company if you fail to comply with any provisions of this license. Upon termination, you shall destroy all copies of the Software and any accompanying documentation. All provisions of this Agreement as to warranties, limitation of liability, remedies or damages shall survive termination.

5. Copyright Notice. The Company and/or its Licensors hold valid copyright in the Software. Nothing in this Agreement constitutes a waiver of any rights under U.S. Copyright law or any other federal or state law.

6. Miscellaneous. This Agreement shall be governed by the laws of the United States of America and the State of Minnesota. If any provision, or any portion, of this Agreement is found to be unlawful, void, or for any reason unenforceable, it shall be severed from, and shall in no way affect the validity or enforceability of the remaining provisions of the Agreement.

7. Limited Warranty and Disclaimer of Warranty. For a period of 90 days from the date on which you purchased Software, the Company warrants that the media on which the Software is supplied will be free from defects in materials and workmanship under normal use. If the Software fails to conform to this warranty, you may, as your sole and exclusive remedy; obtain a replacement free of charge if you return the defective Software to us with a dated proof of purchase. The Company does not warrant that the Software or its operations or functions will meet your requirements, nor that the use thereof will be without interruption or error.

EXCEPT FOR THE EXPRESS WARRANTY SET FORTH ABOVE, THE COMPANY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING AND WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. EXCEPT FOR THE EXPRESS WARRANTY SET FORTH ABOVE, THE COMPANY DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATION REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE.

IN NO EVENT SHALL THE COMPANY OR ITS EMPLOYEES OR LICENSORS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE LICENSE GRANTED UNDER THIS AGREEMENT INCLUDING AND WITHOUT LIMITATION, LOSS OF USE, LOSS OF DATE, LOSS OF INCOME OR PROFIT, OR OTHER LOSS SUSTAINED AS A RESULT OF INJURY TO ANY PERSON, OR LOSS OF OR DAMAGE TO PROPERTY, OR CLAIMS OF THIRD PARTIES, EVEN IF THE COMPANY OR AN AUTHORIZED REPRESENTATIVE OF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL LIABILITY OF THE COMPANY FOR DAMAGES WITH RESPECT TO THE SOFTWARE EXCEED THE AMOUNTS ACTUALLY PAID BY YOU, IF ANY, FOR THE SOFTWARE.

SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MAY NOT ALWAYS APPLY.

ACKNOWLEDGMENT

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU ALSO AGREE THAT THIS AGREEMENT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND THE COMPANY AND SUPERCEDES ALL PROPOSALS OR PRIOR ENDORSEMENTS, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN YOU AND THE COMPANY OR ANY REPRESENTATIVE OF THE COMPANY RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

Exclusive new scenarios

Copyright©1998 Micropose, Inc. All Rights Reserved.

Published by: GW Press
A Division of Game Wizards, Inc.
7085 Shady Oak Road
Minneapolis, MN 55344
www.gwpress.com

GOLD EDITION CIVILIZATION II MULTIPLAYER

**NOW
AVAILABLE!**

Competition
has never been so fierce and
worldwide domination
never so rewarding!



Available on Windows 95. For ordering, visit your local
retailer, call 1-800-695-GAME day or night (U.S. or Canada)
or visit our Web site at www.microprose.com.

MICROPROSE
www.microprose.com

Appendix C

HOW TO USE THE CD-ROM

This book originally came with a CD-ROM. If you purchased this book and there was no CD-ROM, you got an incomplete copy. Go back to wherever you bought the book and ask for a complete copy.

We've built three brand new, exclusive scenarios and included them on the CD-ROM. Let's not waste time talking, then; let's get you playing.

Installation Requirements

You need to have *Civilization II* installed in order to be able to run these scenarios. You also need roughly 3MB of free hard drive space to install all three scenarios.

The scenarios on the CD were built using *Fantastic Worlds*; therefore, you must also have either *Fantastic Worlds* or the multiplayer edition of *Civilization II* installed in order to play them.

NOTE

Installing the Scenarios

Getting the new scenarios from the CD-ROM to your hard drive is simple. All you need to do is copy over the files and put them in the right places. Use whatever file copying program you're comfortable with—DOS, Windows Explorer, or whatever.

The files for the scenarios are stored on the disk in these folders:

Table C-1. Where the Files Are

SCENARIO	FOLDER	FILE
Labyrinth	Labyrnth	labyrnth.scn
Eden	Eden	eden.scn
Dragons	Dragons	dragons.scn

All you need to do is find the scenario folders on the CD-ROM, then copy the onto your hard drive as subfolders of the *Civ II Scenario* folder. That's it. ✕

- 3 Delete the header `@TRIGGERS` and every line after that up to (but not including) the header `@DELETEITEM`.

To make a *game.txt* from *Fantastic Worlds* work with *Conflicts in Civilization*. Follow the same procedure, but skip steps 1 and 3 (just do step 2). Instead of deleting the two sections relevant to event triggers and actions (as in step 3), replace them with the text from the *Conflicts in Civilization* version of *game.txt*.

Events.txt

If you're making a scenario work with the original game, you can just delete the *events.txt* file. It's of no use unless at least one of the scenario packs is installed.

There are minor differences in the valid triggers and actions allowed in *Conflicts in Civilization* and *Fantastic Worlds*. You must remove all the invalid events. All the changes are detailed at the end of the *Fantastic Worlds* manual, but here's a summary. For the events file to work with *Conflicts in Civilization*:

- ▼ The `ReceivedTechnology` trigger must be removed.
- ▼ The parameters `TriggerAttacker`, `TriggerDefender`, and `TriggerReceiver` must be removed.
- ▼ Any `GiveTechnology` action must be removed.
- ▼ Any `DestroyACivilization` action must be removed.
- ▼ Any `ChangeTerrain` action must be removed.

Graphics Files

If you used the editors to make changes to any of the icons or other graphics, *Fantastic Worlds* stored the new pictures in files of the BMP format. (This is a standard Windows graphic file format.) Versions of *Civilization II* previous to *Fantastic Worlds* do not recognize that format, so you need to change those files back to the GIF format.

In this one case, you *must* have the use of a GIF editor. The editors in *Fantastic Worlds* caused this problem, and you can't use them to fix it.

It's not that difficult. If you load the BMP file into your GIF editor, you should be able to use the Save As ... function to change the file back into the GIF format. ✕

NOTE

Labels.txt

The *labels.txt* file was updated quite often during the development of *Fantastic Worlds*, but undoing the changes is only a two-step process.

- 1 Search through the file for the line `SCORING COMPLETE`. Delete everything in the file after that line.
- 2 At the beginning of the file, there is a header for the string heap (`@STRINGHEAP`). On the line after that header, change the number `16368` to `11000`.

If this doesn't work or causes problems (the string heap number is particularly sensitive), there's an alternate method you can try.

- 1 Copy the original *labels.txt* from the *civ2* folder on the original *Civilization II* CD-ROM. (You'll need to give it a temporary name.)
- 2 In that copy, delete all of the text between the lines `@POPUPS` and `SCORING COMPLETE`—but not those two lines themselves.
- 3 Copy all of the text between the lines `@POPUPS` and `SCORING COMPLETE`—but not those two lines themselves—from the scenario version of the file into the copy of the original. Put them where the lines you deleted were.
- 4 Now delete the scenario version and rename the new copy *labels.txt*.

Game.txt

There are three versions of this file, one each for the original game and both scenario packs.

To make a *game.txt* from *Fantastic Worlds* work with the original:

- 1 Delete the header `@PICKMUSICSCENARIO` and every line after that up to (but not including) the header `@PICKMUSICFANWORLDS`.
- 2 Delete the header `@PICKMUSICFANWORLDS` and every line after that up to (but not including) the header `@@CENTAURI`.

Appendix B

CONVERTING SCENARIOS

Even if you don't use the editors, any scenario you create after you have installed the *Fantastic Worlds* scenario pack will not work with any version of *Civilization II* that doesn't also include *Fantastic Worlds*. That's because MicroProse changed some of the files.

This appendix is nothing more than a set of instructions on how to convert *Fantastic Worlds* format scenarios into a form that earlier versions of *Civilization II* can use. You'll never need to convert older scenarios into *Fantastic Worlds* format, because you'll have no problem running them. (*Civ II* add-ons are backwards compatible, so *Fantastic Worlds* can run old scenarios just fine.)

By following the instructions in this appendix, you can convert all of the listed files into a format that works with any version of *Civilization II*. However, the scenario file itself (scenname.scn) is not listed. That's because there is no easy way to convert the scenario file. *Fantastic Worlds* scenario and save files include more data than earlier versions. You can easily reuse all of your subsidiary files, but unfortunately, you must re-create the scenario situation from scratch.

NOTE

Rules.txt

The new version of *rules.txt* has two sets of added lines. All you need to do is remove the extra lines.

- 1 At the end of the units listings, delete the final eight slots—**Test Unit 1** through **Test Unit 8**. Remove the entire line in each case. If the scenario in question used those slots, you have a bigger problem. You'll need to either live without those units or find a way to shuffle them into other slots. Whichever you do, you'll also need to go into the scenario and replace all units of the affected types—and you should make sure that none of the deleted slots is referred to in any events.
- 2 At the end of the listings for advances, delete the final seven slots—**Extra Advance 1** through **Extra Advance 7**. Remove the entire line. As with the extra unit slots, if those slots are used in the scenario, you'll need to shuffle the tech tree around to heal the gap left by the missing advances. Check the events to see that none of these slots is referred to.

Table A-4. Maps and Miscellaneous Other Files

FILENAME	DESCRIPTION
<i>Battle.mp</i>	A map extracted from the Battle of the Sexes scenario (<i>Fantastic Worlds</i>).
<i>Eastus.mp</i>	A map extracted from the American Civil War scenario (<i>Conflicts in Civilization</i>).
<i>Europe.mp</i>	A large (75 x 120) map of Europe, including Scandinavia, the Atlantic, bits of North America and Africa, most of Asia and the Arab world.
<i>Greece.mp</i>	A medium sized (50 x 80) map of the eastern Mediterranean and points east all the way to India.
<i>Mediterr.mp</i>	A medium sized (50 x 80) map of the Mediterranean Sea, including Greece, Italy, most of Spain, northern Africa, and the western Arab world.
<i>Namerica.mp</i>	A map extracted from the New World scenario (<i>Fantastic Worlds</i>).
<i>Pacific.mp</i>	A large map (75 x 120) that includes the Pacific and its islands, Japan, Australia, the Korean peninsula, western North America, eastern Asia, and a bit of India.
<i>Samurai.mp</i>	A map extracted from the Samurai scenario (<i>Fantastic Worlds</i>).
<i>Tutorial.sav</i>	The save file used with the tutorial in the original game manual.
<i>World.mp</i>	A large (75 x 120) map of the entire Earth.
<i>World_m.mp</i>	A medium (50 x 80) map of the Earth.
<i>World_s.mp</i>	A small (40 x 50) map of the Earth.
<i>Scenname.scn</i>	Any file with the extension *.scn is a scenario file. ✂

FILENAME	PLAYED WHEN...
<i>Newbank.wav</i>	One of the player's cities finishes building the improvement in slot 10.
<i>Newgovt.wav</i>	The player chooses a new form of government.
<i>Newonder.wav</i>	The successful completion of a Wonder of the World is announced.
<i>Nukexplo.wav</i>	Any unit with an attack factor of 99a attacks. This overrides <i>any</i> other sounds the unit might be eligible for.
<i>Sell.wav</i>	The player sells off a City Improvement.
<i>Smallexp.wav</i>	Despite what its name implies, this sound is not used.
<i>Spysound.wav</i>	A diplomatic unit (role=6) successfully bribes a unit, investigates a city, establishes an embassy, or incites a revolt.
<i>Stkmark.wav</i>	One of the player's cities finishes building the improvement in slot 22.
<i>Swordfgt.wav</i>	A ground unit that does not fit the criteria for any of the other ground unit sounds attacks. (For the exceptions, see <i>Catapult.wav</i> , <i>Cavalry.wav</i> , <i>Elephant.wav</i> , <i>Fire——.wav</i> , <i>Infantry.wav</i> , <i>Mchnguns.wav</i> , <i>Missile.wav</i> , <i>Nukexplo.wav</i> , and <i>Swr dhors.wav</i> .)
<i>Swr dhors.wav</i>	A ground unit in slot 15 (Horsemen), 16 (Chariot), 18 (Crusaders), or 19 (Knights) attacks.
<i>Torpedos.wav</i>	A naval unit with the <i>Has the Submarine Characteristics</i> attribute attacks.
<i>Volcano.wav</i>	A volcano erupts in any of the scenarios from <i>Fantastic Worlds</i> ; this sound was added for that pack.
<i>Win.wav</i>	The player reaches one of the alternate conclusions in any of the scenarios from <i>Fantastic Worlds</i> ; this sound was added for that pack.

<i>Fanfare6.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fanfare7.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fanfare8.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fire—.wav</i>	A ground unit in slot 24 (Cannon), 25 (Artillery), or 26 (Howitzer) attacks. This sound is normally followed by <i>Medgun.wav</i> , then <i>Largexpl.wav</i> .
<i>Guillotn.wav</i>	During the winning animation, the guillotine falls to herald the demise of another civilization.
<i>Helishot.wav</i>	An air unit with a Range of zero (0) attacks.
<i>Infantry.wav</i>	A ground unit in slot 7 (Musketeers), 9 (Partisans), 10 (Alpine Troops), or 11 (Riflemen) attacks.
<i>Jetbomb.wav</i>	A non-missile air unit in slot 30 (Stealth Fighter) or any higher-numbered slot attacks a city, a ground unit, or a naval unit.
<i>Jetcombt.wav</i>	A non-missile air unit in slot 30 (Stealth Fighter) or any higher-numbered slot attacks another air unit.
<i>Jetcrash.wav</i>	An air unit in slot 30 (Stealth Fighter) or any higher-numbered slot is destroyed in combat.
<i>Jetsputr.wav</i>	An air unit in slot 30 (Stealth Fighter) or any higher-numbered slot is destroyed for exceeding its range.
<i>Largexpl.wav</i>	As the final sound when any naval unit attacks. This is also the final sound when a ground unit that plays <i>Medgun.wav</i> (slots 22, 24, 25, and 26) attacks.
<i>Mchnguns.wav</i>	A ground unit in slot 8 (Fanatics), 12 (Marines), 13 (Paratroopers), or 14 (Mech. Inf.) attacks.
<i>Medexpl.wav</i>	Despite what its name implies, this sound is not used.
<i>Medgun.wav</i>	A ground unit in slot 22 (Armor) attacks. This sound is also played after <i>Fire—.wav</i> (for ground units in slots 23 through 25).
<i>Menuend.wav</i>	At the end of the game setup process.
<i>Menuloop.wav</i>	During the entire game setup process.
<i>Menuok.wav</i>	You click on the OK button during the game setup process.
<i>Missile.wav</i>	Any unit with the <i>Destroyed After Attacking</i> attribute attacks. This overrides any other sounds the unit might be eligible for except <i>Nukexpl.wav</i> .
<i>Movpiece.wav</i>	The player moves a unit.
<i>Mrktplce.wav</i>	One of the player's cities finishes building the improvement in slot 5.
<i>Navbttle.wav</i>	A naval unit in slot 37 (Destroyer), 38 (Cruiser), 39 (AEGIS Cruiser), or 40 (Battleship) attacks. This sound is normally followed by <i>Largexpl.wav</i> .
<i>Neg1.wav</i>	The player tries to do something that is not allowed.



FILENAME	PLAYED WHEN...
<i>Custom2.wav</i>	The unit in slot 52 (Extra Ship) attacks.
<i>Custom2.wav</i>	The unit in slot 53 (Extra Air) attacks.
<i>Crwdbugl.wav</i>	See <i>Fanfare1.wav</i> .
<i>Diesel.wav</i>	A trade unit (role=7) in slot 49 (Freight) arrives at its destination and fulfills one of its functions.
<i>Divcrash.wav</i>	An air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot is destroyed in combat.
<i>Divebomb.wav</i>	A non-missile air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot attacks a city, a ground unit, or a naval unit.
<i>Drumal.wav</i>	Whenever the player has an audience with one of the computer-controlled civilizations, after the fanfare is over, one of the three drum loops plays until the audience is over.
<i>Drumbl.wav</i>	See <i>Drumal.wav</i> .
<i>Drumcl.wav</i>	See <i>Drumal.wav</i> .
<i>Elephant.wav</i>	A ground unit in slot 17 (Elephant) attacks.
<i>Endoturn.wav</i>	Despite what its name implies, this sound is not used.
<i>Engnsput.wav</i>	An air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot is destroyed for exceeding its range.
<i>Extra1.wav</i>	The unit in slot 54 (Test Unit 1) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra2.wav</i>	The unit in slot 55 (Test Unit 2) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra3.wav</i>	The unit in slot 56 (Test Unit 3) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra4.wav</i>	The unit in slot 57 (Test Unit 4) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra5.wav</i>	The unit in slot 58 (Test Unit 5) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra6.wav</i>	The unit in slot 59 (Test Unit 6) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra7.wav</i>	The unit in slot 60 (Test Unit 7) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Extra8.wav</i>	The unit in slot 61 (Test Unit 8) attacks. (This unit slot does not exist unless you have <i>Fantastic Worlds</i> installed.)
<i>Fanfare1.wav</i>	Whenever the player has an audience with one of the computer- controlled civilizations, one of the eight Fanfare sounds or <i>Crwdbugl.wav</i> plays as the audience begins. Each leader is accompanied by the same fanfare throughout a single game, but which fanfare is associated with each leader is determined at random at when the beginning of the game begins.
<i>Fanfare2.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fanfare3.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fanfare4.wav</i>	See <i>Fanfare1.wav</i> .
<i>Fanfare5.wav</i>	See <i>Fanfare1.wav</i> .

<i>People.gif</i>	The population faces for the City Display.	No
<i>Scredits.gif</i>	The background for the credits for <i>Conflicts in Civilization</i> .	No
<i>Terrain1.gif</i>	The bases for all the terrain types, all the special resource icons (plus lots of unused alternates), roads, railroads, irrigation, mining, farmland, pollution, and the dithering plan for adjacent squares.	Yes
<i>Terrain2.gif</i>	The overlays for those terrain types that have multiple tiles (rivers, forests, mountains, and hills), the plan for how these multiple tiles connect, river mouths, and coastlines.	Yes
<i>Title.gif</i>	The opening graphic displayed when the player loads the scenario.	No
<i>Units.gif</i>	All the units in the game and the definition of their shields, including barbarians and extra units. Note that the second scenario pack, <i>Fantastic Worlds</i> , changes the placement of the barbarian chief unit and makes some of the formerly inactive areas active.	Yes

Table A-3. Sound Files

FILENAME	PLAYED WHEN...
<i>Aircombt.wav</i>	A non-missile air unit with a non-zero range in slot 29 (Helicopter) or any lower-numbered slot attacks another air unit.
<i>Aqueduct.wav</i>	One of the player's cities finishes building the improvement in slot 9.
<i>Barracks.wav</i>	One of the player's cities finishes building the improvement in slot 2.
<i>Biggun.wav</i>	When any naval unit attacks, other than those in slots 37 through 40 and any with special sounds (<i>torpedo.wav</i> and such).
<i>Bldcity.wav</i>	One of the player's settler-type units establishes a new city or adds to an existing one.
<i>Bldspcsh.wav</i>	One of the player's cities finishes building a spaceship part, one of the improvements in slot 35, slot 36, or slot 37.
<i>Boatsink.wav</i>	A naval unit is lost at sea.
<i>Catapult.wav</i>	A ground unit in slot 23 (the Catapult position) attacks.
<i>Cathedrl.wav</i>	One of the player's cities finishes building the improvement in slot 11.
<i>Cavalry.wav</i>	A ground unit in slot 20 (Dragoons) or slot 21 (Cavalry) attacks.
<i>Cheers1.wav</i>	There's a one-in-three chance that this plays when one of the player's cities finishes building an improvement that does not have a specific sound associated with it. Otherwise, one of the other two cheers files plays.
<i>Cheers2.wav</i>	There's a one-in-three chance that this plays when one of the player's cities finishes building an improvement that does not have a specific sound associated with it. Otherwise, one of the other two cheers files plays.
<i>Cheers3.wav</i>	There's a one-in-three chance that this plays when one of the player's cities finishes building an improvement that does not have a specific sound associated with it. Otherwise, one of the other two cheers files plays.
<i>Civdisor.wav</i>	The game announces that one of the player's cities is in disorder.
<i>Custom1.wav</i>	The unit in slot 51 (Extra Land) attacks.

<i>Rules.txt</i>	This file effectively defines most of the things you'll want to change for a scenario: the cosmic constants, units, advances, city improvements and wonders, preset civilizations and their leaders, terrain and resources, trading commodities, and the names of the government types, orders, attitudes, and difficulty levels.	Yes
<i>scenname.txt</i>	If you create a text file in the same folder as your scenario and give it the same name as the scenario file (<i>scenname</i>), <i>Civilization II</i> uses that text file as the introductory text for the scenario.	No
<i>Scredits.txt</i>	This file houses the credits for <i>Conflicts in Civilization</i> . Changes to this file won't show up in scenarios, but if you change the copy of this file that is in the <i>Civ II</i> base folder, the changes will show up when you view the credits.	No

There are French and German language copies of each of these text files in the *Civ II* base folder. Those files have the same names, but they do not have the usual *.txt extension. Rather the German files have the extension *.ger and the French files *.fre. If you want to edit the German or French text of the game (or both) and create scenarios playable by French or German speakers (or both), you can edit these files just as you would the text files with the .txt extension. Note that you must provide your own language skills. *Civilization II* does exactly as much to ensure that your use of these languages is correct as it does for the English version—nothing.

Table A-2. Graphics Files

FILENAME	SOURCE OF...	EDIT WITH FW?
<i>Cities.gif</i>	All the city icons placed on the map (and a few unused alternates), plus airfields, city flags, fortresses, and the unit fortification graphic.	Yes
<i>City.gif</i>	The background for the City Display.	No
<i>Editorpt.gif</i>	This is the pterodactyl background used by some of the editors in <i>Fantastic Worlds</i> .	No
<i>Editorsa.gif</i>	This is the samurai picture used as a background by some of the editors in <i>Fantastic Worlds</i> .	No
<i>Editorsq.gif</i>	This is the unspeakable horror background used by some of the editors in <i>Fantastic Worlds</i> .	No
<i>Editoras.gif</i>	This is the astronaut picture used as a background by some of the editors in <i>Fantastic Worlds</i> .	No
<i>Icons.gif</i>	Icons for the City Display, including city improvements, Wonders of the World, pollution potential, gold, food, shields, global warming, research, happiness and unhappiness, and luxuries; also included are the advances icons (by epoch and category), civil disorder, the eight-frame combat animation, and the buttons for window corners.	Yes

FILENAME	FUNCTION	EDIT WITH FANTASTIC WORLDS?
<i>Council1.txt</i>	This is the second of the three High Council text files.	No
<i>Council2.txt</i>	This is the third of the three High Council text files.	No
<i>Credits.txt</i>	The original <i>Civilization II</i> credits are in this file. Changes to this file won't show up in scenarios, but if you change the copy of this file that is in the <i>Civ II</i> base folder, the changes will show up when you view the credits.	No
<i>Debug.txt</i>	This one has assorted error messages and the text that appears in all the pop-up boxes (and a few windows) for the options on the Cheat menu. Frankly, changing this text for a scenario is pretty pointless.	No
<i>Events.txt</i>	This file contains all of the <i>events</i> programmed into the scenario. Note that the original and Mac versions of <i>Civilization II</i> do not recognize or use this file; you must have one of the scenario packs to use events in your scenarios.	Yes
<i>Fwcredits.txt</i>	The credits for <i>Fantastic Worlds</i> are in this file. Changes to this file won't show up in scenarios, but if you change the copy of this file that is in the <i>Civ II</i> base folder, the changes will show up when you view the credits.	No
<i>Game.txt</i>	All of the pop-up boxes and some of the windows in the game—the start-up stuff, error notices, diplomatic messages, win and loss messages, and so on—are in this file. This one is quite useful for scenarios.	No
<i>Help.txt</i>	This is the Help text for the Editors menu in <i>Fantastic Worlds</i> .	No
<i>Labels.txt</i>	This file holds the majority of the miscellaneous text that shows up during a game. (That is, anything not in <i>game.txt</i> is probably here.) You can edit most of this text with impunity, and the changes do show up in a scenario. Note that the second scenario pack, <i>Fantastic Worlds</i> , establishes a new format for this file.	No
<i>Macro.txt</i>	The text version of the documentation for the scenario macro language is in this file. It's pointless to edit this.	No
<i>Menu.txt</i>	This one contains exactly what it sounds like—the text of the menus on the menu bar. You can change the text, but not the function, of the menu options. Note that the second scenario pack, <i>Fantastic Worlds</i> , installs a new version of this file.	No
<i>Pedia.txt</i>	This file contains the text Civilopedia. The multimedia Civilopedia is never available in scenarios, so this is the Civilopedia for your scenario.	No

Appendix A

FILE REFERENCE

This appendix is a reference to all the *Civ II* files you would ever want to modify in the course of building a scenario, plus a few others that can be useful or fun to play with. Note that this is meant *only* as a reference. Please do not use the information presented here as a surrogate for reading the rest of the book. The steps and chapters tell you how to safely modify the files. Remember—always copy the file into the scenario folder and edit the copy (or edit your full copy of the game if you don’t have either scenario pack). Don’t mess with the original files.

Table A-1. Text Files

FILENAME	FUNCTION	EDIT WITH FANTASTIC WORLDS?
<i>Advice.txt</i>	All the advice the advisors give the player (when the Instant Advice option is turned on) is in this file. You can change this text for your scenarios, but unfortunately you can’t change the fact that most experienced players believe the advisors are all idiots.	No
<i>City.txt</i>	Contains all the predetermined city names for the civilizations defined in <i>rules.txt</i> .	No
<i>Citypref.txt</i>	One of the patches for <i>Civilization II</i> added the possibility of this file; it doesn’t exist unless you create it. This file contains your instructions for the Domestic Advisor to use as his Autobuild orders for your cities (and one toggle for the Military Advisor).	No
<i>Civ2fanw.txt</i>	This is the Readme file for <i>Fantastic Worlds</i> . Editing it has no effect.	No
<i>Council0.txt</i>	This is the first of three files that hold the text displayed in the lower portion of the window when the members of the High Council speak. Changing this does not change what they say, only the text that appears.	No

War for Independence

Tribe to rule: Americans

Could win: British

Optimum players: 2 (maybe 3)

Best for: Handicapping an expert

This scenario can be pretty one-sided, unless the British and the French ally against the rebellious Americans—which can only happen if both European nations are ruled by human players. The British have absolute naval supremacy, but unless they can get a serious foothold on the continent fairly early in the game, the Americans will out-produce them in the long war. This scenario is a good way to give an experienced player a handicap against a less skilled opponent.

World War: 1979

Tribe to rule: North Americans

Could win: Soviet Bloc, Chinese

Optimum players: 4

It could happen: Western Europe

The beginning of this scenario is just a colossal nuke-fest. It has to be. If you don't launch your missiles early on (and follow up with paratroopers), someone else will—or they'll use their spies. The Russians have the advantage in that they play first, but the North Americans have position. Their cities are out of range of most of the missiles in the early game (except the Chinese ones), and they have all of South America to spread into. In the long game, which this turns into after the birds have all flown, usually turns into a naval and submarine war between the Americans, who come in trying to mop up what's left after the initial barrage, and whoever comes out on top in Eurasia. Settle in; this one starts with a bang, but winning can take some time.

X-COM Assault

Tribe to rule: The Aliens

Could win: X-COM

Optimum players: 2

Strong point: Fast and furious!

This scenario is perfect for a two-player slugfest. Research is pointless, and no one can make new cities, so it's entirely a matter of ground tactics. The Aliens have a serious advantage, so you and your opponent must decide who's better, and have that person lead X-COM (it's called "handicapping"). The key to an X-COM win is production; consider disbanding a few units inside cities to hurry completion of some big guns. The key to an alien win is simple—overwhelming force. ☒

Mongol Horde

Tribe to rule: Chinese

Could win: Just about anybody

Optimum players: 2 to 7

Notable feature: Very well balanced

This is one of the most balanced scenarios there is for multiplayer games. Any tribe, if ruled well, can expect to win—except maybe the Mongols. (I chose the Chinese as the best tribe because, in addition to being as big and strong as anyone, they also have less enemy cities nearby and more access to the sea.) There's no special strategy for this scenario except to keep the limitations of your government in mind. Everyone but the despotic Mongols is in Monarchy, and will stay that way. Government switching is not allowed.

New World

Tribe to rule: Chinook

Could win: Anyone

Optimum players: 2 to 6

Don't even try: Africans

This is a nicely balanced scenario for multiplayer games. Everyone has a nice location and a good batch of nearby resources. I only chose the Chinook because they start farthest from other tribes. The Mohawk and Adena begin too close to one another; early war is a distinct possibility. The same goes for the Aztec and Maya. The strategy for winning this one is the same as in any other multiplayer *Civ II* game.

Samurai

Tribe to rule: Fujiwara

Could win: Koreans

Optimum players: 2 to 5

Fun factor: Byzantine feudal diplomacy

If you play the Fujiwara, you'll win unless everyone else gangs up on you, but it's not much of a challenge that way. I suggest playing this scenario as it was designed to be played; get four players together, and each of you take one of the warring clans—Taira, Shimaru, Minamoto, and Hojo. If you have a fifth wheel, give that person the Koreans, but watch out; they've got plenty of room to expand, which the Japanese tribes don't. Expect Gunpowder to show up early and spread rapidly—it can only be in one of two places. This game can be really interesting, and the diplomacy gets pretty byzantine as the four of you vie over a bunch of tiny islands.

Master of Magic

Tribe to rule: Chaos

Could win: Anyone

Optimum players: 5

Big hint: Chart the Techs and Units

This scenario is pretty well balanced for multiplayer action, but before you even think about engaging other humans in this one, you should play it solo and map out the Tech Tree. Also, figure out which units are useful and which are wasteful—anyone who wants to win will have done so. Chaos is the most likely to win only because the red tribe's units are inherently more damaging than everyone else's. This game really is a toss-up.

Master of Orion

Tribe to rule: Klackons

Could win: Almost anyone

Optimum players: 3 to 6

Dark horse: Humans

The Klackons have Fundamentalism without any of the research penalties. Yes, the Elerians have the Great Library, but that could be captured. Yes, the Elerians know the map, but so does anyone who's played the scenario before. The Humans have two Wonders that have no effect on human players. The Gnomes get lots of income, which means they can win the game in the finest tradition of *Civilization*—by bribing the Antaran units. Everyone knows they always appear at (40,40). The only question is: When?

Midgard

Tribe to rule: Goblins

Could win: Anyone

Optimum players: 7

Tough row to hoe: Humans

The Goblins grow fastest, and they start off with cities on the two largest land masses. It's true, however, that a skilled player can win with any of the tribes in this scenario. That's part of what makes it so much fun. The Stygians have a definite advantage in units, but as any experienced *Civ II* player knows, that can be overcome. Just keep in mind that if it's illegal to have a unit appear on a (Entrance to Underworld) square—there's a city or a unit already there—that unit will not appear.



Jihad

Tribe to rule: Byzantines
Could win: Persians

Optimum players: 3 to 6
Not likely: Arabs

This is one of those scenarios that's a whole different game with more than one human player. First of all, the Arabs don't have a chance unless they're allied with the Byzantines, which gives them a little time to grow. Starting with only two cities is a serious handicap, but if you can stay alive, being the only civilization with Fundamentalism available turns into a big advantage. The Byzantines aren't much fun to play, unless: (A) all the other tribes gang up against you, or (B) you just want to win and don't care whether it's a challenge. The Persians need to firm up their defenses fast. If they do, and the Byzantines are kept busy elsewhere for a while, the Persians have some chance—mostly due to the fact that they have the most room to expand.

Jules Verne

Tribe to rule: Continentals
Could win: Not the Exotics

Optimum players: 5
Fun to role play: Secret Evil Society

The multiple possible starting situations make this one interesting, but as a multi-player scenario, it's not as well balanced as some. Expect to engage in fights over the quests, then expect everyone who doesn't gain the quest advances to steal them as soon as possible. Also, look for someone to quietly wait at home until all of the opponent's units are out pursuing quests, then pounce on the poorly defended cities left behind. Most of the world is sparsely settled; a smart player can use this to advantage. Remember that the From the Earth to the Moon "research win" doesn't really end the game.

Mars

Tribe to rule: Americans
Could win: Anyone

Optimum players: 5
Tough row to hoe: Martians, Ukrainians

Like in the Age of Reptiles, every civilization on Mars starts off with the same setup, except for location and resources. The main problem here is the limitations on irrigation, which seriously hampers expansion. The Americans and French have the best terrain for expanding their watered lands, and the Americans also have the most defensible capitol (at least until the Russians build boats.) Before you get into a multi-player game with this scenario, chart the Tech Tree and plan your route through it. Your best tactic might be to hit the nearest tribes early, while they're still weak, then settle in for a long, slow war with the others.

The Great War

Tribe to rule: Russians

Could win: Central Powers,
English, French

Optimum players: 2 to 5

Dark horse: Ottoman Turks

This scenario is great for multiplayer games. It's got all the necessary ingredients; the combatants are evenly matched, in close proximity, and to win, each requires a different strategy. So why did I choose the Russians? I had to choose someone, and they aren't surrounded or cornered from the start; the Russians have all of Asia to spread into. The English have some great hiding places (America and Asia), and they're probably the second best—or first under the right leader. The Central Powers have a heck of a hard time winning (they're surrounded by enemies, after all), but they're just incredibly fun to play. They're always in the middle of everything, and the game never gets dull. Don't bother playing the Americans or Italians; they're too small to be fun. One hint—notice that Asia, Africa, Spain, and Scandinavia are practically empty of cities. One more—there's no pollution in this one; guilt-free nukes!

Ice Planet

Tribe to rule: Ventry

Could win: Anyone

Optimum players: 3

Interesting: Trying to invade others'
Tech Trees

The Ventry have an advantage in both number of cities and available territory, and if they use it well, they can contain the Kydextrians long enough to destroy them. This can give the Galaxials time to get their feet under them, though, and lead to a protracted battle. Any way you look at it, this looks to be a long game. Only the Galaxials are supposed to be able to build a space ship, but that's only because no one else's Tech Tree includes the proper advances. If anyone else gets the right techs from them, they can opt for the (extremely difficult) research route, too.



American Civil War

Tribe to rule: Confederates
Could win: Federals

Optimum players: 2 (maybe 3)
Spoiler: Kentuckians

This is another nearly pure military situation. Research can have an effect, but only if the game goes on longer than usual. The longer the war goes on, the more the Federals' production advantage will matter. The Confederates are stronger at the start, and if they don't take advantage of it, they'll usually lose in the long war. One of the big factors in this scenario is that the AI is easy to beat. Expect any human player to be much tougher, and you won't fall into the trap of underestimating your opponent. Generally, the Confederates can move Stonewall Jackson through Harrisburg and take Albany early on. The Federals best bet is to move quickly on Charleston. Both sides will want to take Kentucky before they can get their defense built up.

Atlantis

Tribe to rule: Europeans
Could win: All but Lemuria

Optimum players: 2 to 6
Challenge: Escaping Atlantis

This one's a hoot to play. It's just like a normal *Civ II* game, except that one of the civilizations has a random expiration date, and there are some interesting modifications to the Tech Tree. The Europeans have the advantages of multiple cities and being near both the Atlantean and Lemurian research trading opportunities. Anyone can win this one (except the Lemurians). If you think you're a real expert, try ruling the Atlanteans while other humans try to keep you from establishing a beachhead anywhere off your islands. They know what's coming as well as you do, but no one knows exactly *when*.

Crusades

Tribe to rule: Turks
Could win: Almost Anybody

Optimum players: 4 or 5
Fun factor: Diplomacy really matters

The Turks start off in control of Jerusalem and the Great Library, plus they can use the Assassin unit. This gives them an edge, but not an overwhelming one. The Byzantines have trouble fending off attacks from both sides, but their central location also gives them control over the most direct access into Asia Minor. If they can make a dependable alliance with a nation on any side, they become a tough nut to crack. If the European nations fight among themselves (as human players tend to do), the Turks can take their time killing off Egypt, then sweep through Europe after everyone has exhausted their resources. The overall lesson: make alliances early, and stick with them until the opposition is on the ropes (or gone). *Then* turn on your allies, not before.

Alexander the Great

Tribe to rule: Persians

Optimum players: 3 or 4

Could win: Macedonians, Thracians

Long shot: Independent Greeks

This scenario is set up as a purely military situation; don't bother with research. The Companion Cavalry and the Immortals are nice units. If you've got them, don't abuse them and they'll serve well for a long time. Watch out for bribery attempts, unless you don't have one of those units—in which case, you should be trying to bribe them away. Try to use the AI civilizations as allies to keep the other players busy. The Persians are the strongest at the start, but they're also in the worst position—surrounded on all sides by enemies. The Independent Greeks are tougher than they seem; those two island cities are a nice ace in the hole. Down south, the Egyptians are easy pickings for whoever wants to control the Pyramids. The Indians have Marco Polo's Embassy, which generally turns out to be a useful source of intelligence.

Alien Invasion

Tribe to rule: Hodads

Optimum players: 3 or more

Could win: Humans

Main problem: Hodads too powerful

If you agree ahead of time that no one will play the Hodads, this scenario can be a fun multiplayer game. Otherwise, everybody must gang up on whoever's controlling the aliens just to prevent a rout. Regardless of who you rule, forget the dark map—every player will have played the scenario before and knows where the cities are. Human tribes need to research like mad to achieve parity with the Hodads. Share advances whenever possible, just so long as no one gets to the TIGUR before you. Don't forget to break up those railroad networks to slow down invaders. After the aliens are gone, it's every civilization for itself. Watch out for players who make secret alliances with the aliens—you know how people are.



Age of Napoleon

Tribe to rule: French or Russians **Optimum players:** 2 to 7
Could win: English or (maybe) Turks **False hope:** Leonardo's Workshop

The French start out with a lot cities and of forces—and Leonardo's Workshop (in Milan). The Russians have good position and lots of room to expand. The English have an awesome navy. The Turks have plenty of space, and they're far from everyone. On the other side of the coin, the Austrians, Prussians, and Neutral Alliance are in a pickle—trapped in the middle of the map and not as strong as their neighbors. They're doomed unless they can make some trustworthy alliances. Naval strength is important in this scenario, but it won't do anyone any good in Russia or the eastern region of the map. Research is possible in this scenario, but it's costly—which makes Leonardo's Workshop less important than usual. Don't bother; just duke it out.

Age of Reptiles

Tribe to rule: Aalu **Optimum players:** 2 to 7
Could win: Anyone **John says:** Dinosaurs are fun!

In this scenario, every civilization starts off with the same setup, with the major exceptions of location and resources. Because of this lack of a pre-existing situation, it's pretty well balanced for multiplayer games. (I chose the Aalu as the best to rule because they start alone on an island, which normally gives them time to build up before engaging in a protracted war.) The main thing to remember is that many of the dinosaur units are not particularly useful in the military sense. Before you embark on a multiplayer game with this scenario, chart the Tech Tree and plan the quickest route to the useful units—especially the Triceratops unit, which is the most useful during the middle game. The Earthworks wonder is a little too powerful, too.

The main point is that if you don't play nice, you'll soon find that you aren't allowed to play at all. If you're more concerned with winning than with enjoying the game, you won't have fun, and neither will the folks you're playing with—and they'll let you know it.

Scenario Quickies

None of the MicroProse scenarios was designed to be played as a multiplayer game. That is, they're not balanced for multi-human play. Every one has an optimum number of players—and one civilization that's most likely to win when ruled competently.

Needless to say, one could fill books on these scenarios. Here I'm just going to breeze over the important points to remember about each scenario. Given that head start, you should be able to design your scenarios to provide maximum multiplayer fun.

After the Apocalypse

Tribe to rule: Saurians

Could win: Almost anyone

Optimum players: 3 to 5

Less than likely: Kamikazes, Serene Lights

The way territory is meted out at the beginning of this scenario is a complete mess, which makes it interesting. The strongest tribes, the Mutants and Eradicators, are set up to get in a war right from the start. Everybody else is represented nearby, with the exception of the weakest tribes—the Serene Lights and Kamikazes. The Saurians have all of North and South America to spread into, which gives them an advantage both in territory and distance from the other tribes. A strong alliance between the Kamikazes and Serene Lights could result in a dark horse victory.

Age of Discovery

Tribe to rule: Neutral Alliance

Could win: True Peoples

Optimum players: 2 to 5

Fun factor: Destroying the imperialists

In a multiplayer contest, this scenario is turned on its head (and so is history). The usual imperial protagonists—the French, English, and Spanish—don't have a chance against a well-run Eurasian or Native American empire. It's just a matter of the numbers; both of those tribes have lots of cities and plenty of area for expansion. The contest starts in earnest after the Europeans have been eliminated, and it's a typical *Civ* end-game. The Tech Tree is not limited in this scenario, so sooner or later, it's going to turn into an America versus Asia nuke-fest.



false conclusion. Take the person's personality (that is, the person's "king" personality as well as what you know of the real person behind the "leader") into account when deciding what to believe.

- ▼ **Choose your long-term allies well.** If you and another player have the same strengths and weaknesses—you both do well at research, for example, but are no good at economics—you might not be the best team. Find an ally whose skills complement yours, and trade benefits. This allows you both to specialize to some degree, and can effectively double your progress.
- ▼ **Last, but not least, be a smart space explorer.** If you ever launch a spaceship, be sure to have a hideaway city squirreled away in some remote region. Then, as soon as you've launched the colony ship, sell your Palace. When you have no capital (no Palace), other tribes must destroy all of your cities to strand your spaceship. Otherwise, they only need to waste your capital city.

You can't do anything to prevent this sort of thing, but fear not. Multiplayer games are self-policing. Here are a few examples noted from experience with *CivNet*, other multiplayer games, and real life:

- ▼ **Be prepared to compromise.** If someone (or some team) consistently wins, other players can and will cooperate to even the odds. This can take the form of refusing to play unless the champions agree to a handicap—ruling a tribe that has a disadvantage, for example. Players interested in an enjoyable game will not balk at this sort of agreement.
- ▼ **Sleazeballs get a reputation—fast.** Players who consistently use underhanded strategies to gain an advantage will find themselves shunned during games. No one will engage in negotiations with them, for fear of being taken advantage of in some scheme. Trust is easy to lose and very difficult to regain.
- ▼ **Assume that others are talking about you.** Just as a civilization's reputation has an effect on the way AI tribes choose to deal with that ruler, a player's reputation affects the way other humans deal with him or her in the game. Don't for a minute fool yourself that the other players aren't talking to each other. A person with a reputation for sticking to alliances finds that those agreements are easier to negotiate. Dishonorable rulers create a suspicious, dog-eat-dog environment that they are then forced to live in.
- ▼ **No one wants to play with irritating people.** Players who gloat, hurl insults, or make a habit of being obnoxious don't get invited into games. Hosts make a point of excluding them.

General Tactics

Humans are clever. No matter how well balanced you make a scenario, someone will find a way to unbalance it in his or her favor. That's the point, after all—to win (while having fun). To further confound your efforts to make it fun for everyone, multiplayer games add a couple of extra factors—cooperation and manipulation. Here are some points to consider in a multiplayer *Civ II* game:

- ▼ **Human players are more gullible than the AI.** If you're a smooth talker, you'll find that human players are much more manipulable than the AI. Just as it does in real life, lying can bring you a temporary advantage. Just beware the backlash when your dissembling is later uncovered.
- ▼ **Every ruler has fears you can play on.** With some exceptions, no one truly *trusts* the tribe next door. If you can convince two of your neighbors to go to war with one another while you remain neutral, you gain time and the advantage of having both of them waste resources in a protracted conflict. Any other player's loss is your gain.
- ▼ **Some players believe everything they read.** For some reason (human nature is odd), anonymous chat broadcasts are more likely to be taken as fact than chat messages from known sources. If you're spreading disinformation, this can be one of your best tools.
- ▼ **Alliances need not be announced.** If you have a formal alliance with another player, you gain the advantage of being able to move freely through each other's territory, but other players can become aware of the treaty through the usual means. If you agree to cooperate but do not form an alliance (in fact, you could be, technically, at war), your team gains the advantage of secrecy. For example, if you're "at war" with your teammate, another tribe might include you in the planning of a scheme directed at your ally's empire. Not all the espionage takes place on the map!
- ▼ **Of course, formal alliances have their darker side.** If you notice an "ally" positioning units in strategic locations inside your territory, he's up to no good. Consider a strongly worded warning or, even better, a pre-emptive strike.
- ▼ **Pay attention to what other players say.** Even when there's no official transfer of information in a negotiation, you can learn a lot from what other players say—and what they avoid saying. Whole books have been written on this topic, so I can't cover it in a paragraph. The lesson is this: study what others say carefully, and remember that they might be intentionally trying to lead you to a



MULTIPLAYER STRATEGY

I know; I said right at the beginning that this wasn't a strategy guide, and that I wouldn't tell you how to win. So why include a chapter on multiplayer *Civ* strategies? Among other reasons, because it will help you to design better scenarios.

You don't *need* to know how to beat the AI to be able to design a fun scenario, but the situation is different in a multiplayer game. If there is an imbalance in your scenario—if one tribe has an advantage, for example—players will find it and use it in a multiplayer contest. As a designer, you should be aware of the possibilities—and as a player, it won't hurt you to know more than the next guy.

As always, the best advice is to test the scenario *a lot*. Play it yourself, and find others to play it and give you feedback. Game balance problems are not always things you can put your finger on easily. You're going to spend some time playing, tweaking numbers, changing costs, moving units around, and so forth—and as I've seen constantly with other games that allow you to create your own levels, scenarios, and what have you, you'll find that players will find ways to abuse your set-up that you never thought of. Plugging these holes is part of the process. Enjoy it! ✂

Here's one hint that's just common courtesy: Always announce your "utility tribes" to the players in the introductory text. No one will have any fun at all playing them in a multiplayer contest.

Balance

The most important factor is game balance—make sure that it's not too easy for one civilization to win, and also determine that it's not impossible for anyone to win (among other factors). I can't really give you any specific advice on how to accomplish this. Every scenario will have its own trouble spots. Here are a few of the most common imbalances we saw during testing of the MicroProse scenarios:

- ▼ **One civilization's territory is more fertile.** You might be surprised at how much of an advantage productive real estate gives a tribe. If this is balanced out by a disadvantage in some other area, you shouldn't notice the imbalance. If not, it's something to fix.
- ▼ **Private techs are too useful.** If you permit only one tribe to research Mobile Warfare, for example, there's no contest—the guy with the tanks will win. Also, watch out for making private tech out of any advance with an unchanging effect (Automobile's effect on pollution, for instance); it gives one tribe too much control over the progress of the entire game.
- ▼ **Any unit is too strong.** Even if everyone has access to it, an overly powerful unit is an unbalancing factor. There should be a punishment for using it (like there is for Nuclear Missiles), an excessive cost to build it, a built-in limitation (air units' range and inability to take cities work nicely), or some other factor that mitigates the unit's strength.
- ▼ **Tech proceeds at the wrong speed.** You'll often need to fiddle with the tech numbers several times to get it right. If advances come too quickly, tribes run into Future Tech too early, and the game bogs down. If they come too slowly, things can get mighty dull (unless science is cut out of the scenario intentionally).
- ▼ **Events have unshared effects.** Be really careful with your events when designing multiplayer scenarios. If they have greater effects on some civilizations than on others, it's surprisingly easy to (unintentionally) give a clever human ruler an advantage. Especially be wary of changing terrain, creating units, and manipulating players' money. (Needless to say, you shouldn't be using DESTROYACIVILIZATION at all.)



- ▼ **Good things come in small packets.** One way to speed up the pace of a game is to minimize the amount of data that needs to be transmitted each turn. The programmers have done most of the work for you in that area, but there are a couple of steps you can take to make the communications go faster. Designing your scenario on a small map is one; the size of the world puts a natural limit on the number of cities and units that are practical in the game. Making the map mostly Ocean is another way to subtly rein in the more prolific players.
- ▼ **Think war.** The best *Civilization* players balance diplomacy with the occasional *blitzkrieg*, but in a multiplayer game, there's a strong tendency toward fighting. Make sure that your scenario is a fun place to stage a war. Build in geographical features that will make conflicts interesting—natural barriers, strategic bottlenecks, and that sort of thing. Then place the cities and units to create a challenging military situation for everyone involved.
- ▼ **Leave room to expand.** Unless your scenario is a straightforward slug-fest (and sometimes even then), you should leave some areas of your map empty of settlements. One of the major components of the gameplay in *Civilization II* is expansion. A smart player can use empty territory to great advantage, but only if you put it in the scenario.
- ▼ **Consider alliances.** When you've finished the first pass at designing your scenario, sit back and imagine all the likely ways players could team up against each other. You can't prevent alliances, but by taking a good, hard look at the possibilities, you can think of ways to make the scenario more interesting. If you're creative, you might be able to design the game to encourage certain alliances and discourage others.
- ▼ **Build in challenges.** Some *Civ II* players are better than others. If you want players of different levels of experience to be able to play one another and all enjoy themselves, consider making the game tougher for certain tribes and easier for others. (This can be as simple as putting one tribe's territory in the middle of all the others.) This allows expert players to accept a handicap in a game against less skilled opponents. Just be careful not to make the scenario too unbalanced, or it won't be fun for anyone.
- ▼ **Give every tribe a personality.** This is important enough to say twice. *Civilization II* is a strategy game, but even so, part of the fun is playing the role of the ruler of a particular civilization. Just as you should when making solo scenarios, you should give each tribe its own character. At the very least, give every tribe in the game one or a few private units and advances. Just don't give any one a big advantage.

Some Guidelines

If you want to build scenarios that will be interesting and fun as multiplayer games, you don't have to do anything special. Almost any scenario that's interesting for a single player will be even more fun for two or three players. Remember these two guidelines from Step 1: Getting Started:

- ▼ **Have at least three protagonists**, two of which are evenly matched. Scenarios with two civilizations are practically impossible to balance.
- ▼ **The protagonists should not be identical.** Each protagonist must have something in its identity that easily distinguishes it from all others.

“Protagonists” means civilizations powerful enough to be active participants—tribes that have a good chance of winning. If there are at least three of these, then you have a scenario that should be fun for at least three players. Making them different gives each player some kind of individual identity. (If the players wanted to be all the same, they would play a random start game, not your scenario.)

Every scenario will have an optimum number of human players—the maximum number who can play and still all enjoy the game. For example, in Mick's American Civil War scenario, the optimum is probably two. In *After the Apocalypse*, however, seven players could probably have fun. The number depends entirely on how many civilizations there are that have a reasonable chance to defeat all the others. Not too many people would get any enjoyment out of playing the Africans in the New World scenario, because they're quite a small empire, and they get wiped out partway through by an event.

So what do you need to watch out for if you want more than three players to have fun playing your scenarios? Here are a few concerns, though I'm sure this is not an exhaustive list—not by a long shot:

- ▼ **Time is of the essence.** This might be the single most important thing to remember when you're building multiplayer scenarios. Time spent on-line costs money, and not everybody wants to sit there and play the usual hundred-hour *Civ II* game when they're paying by the hour for a connection. A multiplayer scenario should be fast paced and have a definite time limit. (If the players choose to, they can always continue playing after the time has expired.)



DESIGNING MULTIPLAYER SCENARIOS

MicroProse's multiplayer edition of *Civilization II* makes it possible to play any scenario as a multiplayer game. Any scenario that works with the original game or either scenario pack should run just fine. Note that I said it will *run*, not that it will be fun.

When the designers at MicroProse built the first two batches of scenarios, they didn't have multiplayer in mind. Still, those games should be fun for two or three players—it's not *necessary* for a scenario to be designed specifically for multiplayer use in order for it to make for fun multiplayer games. You, however, are not bound by the limitations we at MicroProse had to face; you can build multiplayer scenarios *now*.

PART III

Multiplayer Civilization II

This part goes under the skin of the multiplayer addition of *Civilization II* with tips and strategies from the game's designers.

Just about the time the last few icons were rolling in and I thought I could put the scenario to bed, we got a new executable and I found out that there was a new event action called *ChangeTerrain*. The next day (I had to add the new events stuff to the manual first), I started playing around with it and came up with the land bridge idea. That meant changing the map (to put the Aalu on an island) and significantly changing the play balance of the scenario. Back to QA!

After all the testing and fiddling and all, two last minute fixes didn't even make it into the released version. We found out too late that the Velociraptor and Dimetrodon icons were reversed. Also, the Branchiosaurus icon ended up in the wrong location; it's in the bottom left, when it should be one position to the right. (We've released a patch that fixes them, but fixing them yourself would be good practice.) Oh well, nobody's perfect—and it's still fun. ✂

The Age of Reptiles

If you've read the rest of this book, you've already put up with me using this scenario as an example for just about everything. What can I say—I know the scenarios I designed best, so they naturally come to mind when I'm trying to think of examples.

I'd been working for a couple of weeks on the map for the Mars Now! scenario, but I got bogged down. If you've never tried it, you don't know how tedious it can be, making a map square by square. So I started looking for a distraction—something more interesting to do. I also knew Mick was hoping we could come up with a few more in-house scenarios.

First, I tried something that had been bouncing around in my head for a few days. I took the map from Mick's Samurai scenario and modified it a bit, then turned the last ten units in *rules.txt* into monsters from a certain series of low-budget Japanese monster movies. After a few hours, I had a rough but playable scenario I called "Tokyo Trouble." The only problem was, every monster in the scenario was trademarked by somebody. We certainly weren't going to release *that*, so I had to shelve it.

Designing the monsters had gotten my brain going, though. If I couldn't use those fictional monsters, why not *real* ones? I started making a list of dinosaurs right then and there. I'm embarrassed to admit, it was a short list. Several library books later, however, I had everything I needed, including a rough timeline and lots of color pictures. The colors are mostly guesses, of course, but paleontology includes some room for speculation.

I had a lot of fun naming all the cities, with some help from Anne Stone and a thesaurus (which is not a dinosaur). We put in enough names for all twenty-three civilizations, plus barbarians, even though the scenario only uses the first seven. A lot of them are gibberish, but many are actual English words. (If you don't believe me, look some up.)

The artists at MicroProse's Texas studio and Jim Crawley did a great job on all the icons. Everybody loved the Elvisaurus. Haven't seen it? Take a close look at the Entertainer icon.

When I thought it was done, I shot it off into testing. Right away, QA started complaining because they couldn't pronounce the names of the dinosaurs. It made meetings fun. More serious was the fact that cities weren't growing past size 4, but were able to pump out units every few turns. We reworked the terrain output numbers several times before everyone was satisfied.



It took weeks to make the map. I made a large all-desert world, then used mountain and ocean squares to mark it off into tiny, 10 degree by 10 degree squares. (I had to fudge a lot near the poles.) After that, it was a simple matter of matching every square to the corresponding piece of the Martian surface depicted in Moore's maps, deciding what sort of terrain dominated the area, and placing that terrain in the Map Editor. (Somewhere in the middle of the process, I got bored, took a break, and made the dinosaurs scenario.)

When Mick saw the partially completed map and my plan for the rest of it, he talked me into building a scenario. Naturally, I went right out and got some books. I don't mind admitting that most of the good ideas in the scenario came from Robert Zubrin's *The Case for Mars* (The Free Press, 1996), and other books too numerous to mention, both science fact and science fiction. Mapping out the tech tree was surprisingly easy, but coming up with units was tougher. To avoid making everything far too easy, I had to assume that the colonies had lost contact with Earth. Otherwise, the whole map would be known by satellite reconnaissance, every civilization would know all about the other ones, and research would progress quickly beyond what the *Civ II* engine could reasonably handle.

So here I was with seven advanced cultures, made primitive by a lack of resources rather than technical know-how. While the artists were busy bringing my somewhat bizarre (but all scientifically feasible) units to life, I sat and played the scenario. It crashed every time.

This was a surprise. I'd played it several times before, while testing out the balance of production from the terrain types, testing units, and so forth, and the game had never locked up like that. After some fiddling, I realized that I'd never played more than ten or twenty turns. The Mars scenario crashed, consistently and repeatedly, on turn 32.

As it turned out, we had run into a flaw in the Analyze Map feature of the Map Editor. That particular tool (covered in Step 2: Create Your Map) doesn't check to see if your world has enough Ocean squares. There are a few AI functions that depend on being able to find an ocean, and I had bewildered them by including no Ocean squares at all. That's why there are a few Watershed (Ocean) squares near the poles when the scenario opens; I had to add them so that the AI would agree to play along.

Then we went into QA with it. That's another story.

Psilons got SETI to represent their research bonuses. The Sakkra got the Pyramids to duplicate their fast population growth. The Elerians do not have a wonder, but instead have the entire map revealed to them at the start of the game.

This being the second scenario I created, I ran into fewer problems. I originally intended to have missile units defined as air units with a 2 or 3 square range. I made most ships able to carry a few of them. *Civ II*, however, does not allow ground units to carry air units, so this whole idea had to be scrapped when I switched from a sea to land map. I also attempted to have Antaran units appear as Barbarians. To ensure that they did not appear too early in the game, I used events macro commands to program in a small chance that they would appear after turn 100. I found out later from Mick Uhl, who was invaluable throughout this whole process, that you can't change the name "Barbarians", and if the Barbarian units created are not near a unit or city of some other civilization, they usually just disappear. The chances of that weren't very good, so the Antarans fell by the wayside.

John Possidente

As I'm sure you've figured out by now, I designed a couple of scenarios for *Fantastic Worlds*, too. I've been at MicroProse for almost six years, and I've had opportunities to help out with game designs here and there, but this was my first chance to do the whole thing myself. It was a lot of fun.

Mars Now!

This scenario started out as an idea for a map. I've had an interest in space exploration for as long as I can remember, and one day I was looking up something in Patrick Moore's excellent *The New Atlas of the Universe* (Arch Cape Press, 1988) and I was struck by the detailed maps of Mars. This was before we knew we'd be making *Fantastic Worlds*, but I'd been playing around with the *Civ II* Map Editor, toying with the idea of creating a scenario just for fun. As soon as I saw the level of detail in Moore's maps, I knew they'd make a spiffy *Civ II* map.

Of course, I'd have to completely remake all the terrain types. The original "placeholder" terrain icons I made were absolutely awful. The final ones are better (thanks to Barb and Betsy)—in fact, the Cratered, Permafrost, and Canyon terrain are derived from actual NASA photos of the Martian surface.



Master of Orion II

The Master of Orion II (MoO2) scenario presented a completely different set of problems. I had a tech tree already, but the map structure for the original *Master of Orion II* is completely different from that for *Civ II*. The tech tree I used pretty much as is, except that I matched the number of technologies and I cross-linked the branches to ensure a broad technology base. Without the cross links, there would be nothing to stop a civilization from repeatedly researching the same branch to get deep into the tree very early in the game. Both *MoO2* and *Civ II* use an increased research cost for later technologies to accomplish a balancing effect, but in different ways.

The other problem with the technology tree was in the different ways that units are handled. In *MoO2*, new ship components—weapons and special devices—are invented, then the player uses those to create more powerful units. To convert this to the *Civ II* model, I made several tech categories for each size class of ship. Thus, we have Cruiser I early in the game and, after several more technologies are researched, Cruiser II becomes available with better stats.

The map proved to be a more difficult problem. My first attempt was to make a group of around 15 planets—each consisting of approximately a 5 x 5 square of land—all floating in an ocean that resembled space. This let me have a few ground units to fight on planets, and the spaceships were all naval units. Unfortunately, the *Civ II* AI was unable to handle a world consisting entirely of such small continents. The AI civilizations tended to just sit on their initial planet and not expand. After a great deal of thought, I came to two realizations: the difficulty with the AI was getting them to colonize across water, and ground units are not necessary to a MoO2 scenario.

This allowed me to reverse my initial design and create a map consisting completely of land, except for some ocean around the edges to keep the AI happy. I made all of the space ships ground units, and I eliminated ground troops entirely. I made empty space generally worthless for production and gave planets very high production values. This made it almost impossible to start cities anywhere but in star systems. The AI performed well on this map, and the loss of ground troops had no serious impact on the game.

I used wonders and governments to give each of the races in the game special abilities that roughly corresponded to their abilities in *Master of Orion II*. The Darloks got the Great Library to simulate their espionage abilities. The Gnomes got Adam Smith's Trading Company and a bonus of 10 gold a turn from an event. The Humans got the Eiffel Tower and United Nations because of their diplomatic abilities. The

player build units instead of summoning them were the only changes necessary. Pat Owens created the technology chart before I became involved in the project, so all I had to do was actually implement the scenario. This was my first exposure to the many pitfalls waiting for a designer of *Civ II* scenarios.

The first problem I ran into was trying to make certain portions of the tech tree (Life, Death, Chaos, etc.) available to only one civilization. This is not a feature that's supported by the *Civ II* tech tree structure, and I didn't yet know about Mick's method. I created a dummy advance to give each tribe that was a prerequisite for their particular technology branch. This worked perfectly, until I found that the AI rulers would happily trade this special technology. I didn't actually fix this problem, but I did increase the value of the special technologies to the point that the AI would only trade them if offered another civilization's special technology.

Flying units were another problem. *Civ II* is designed to have airplanes that must land after a certain number of turns. This did not make sense for dragons and such, so I gave them unlimited range. Unfortunately, *Civ II* assumes that any air unit with unlimited range is like the Helicopter and takes damage every turn that it's not at a landing site. There proved to be no good solution to this problem either, so players will notice their dragons taking damage as time goes by.

Unit sound effects can be a big problem in designing *Civ II* scenarios. The rules that govern which sound file each unit uses when it attacks are buried deep in the code. Among other things, the ordering of units in the unit slots is very important. This means that there are a very limited number of sounds available, all the units that are meant to use the same sound file must be placed in the correct slots in the *rules.txt* file. I managed to get appropriate sounds for most of the units, but by necessity, a few are using sounds that are only approximate.

There were a few other minor problems that I ran into. I thought that it might be possible to make units invisible by giving them the submarine special abilities. Unfortunately, units with this ability may not attack ground units, so I had to scrap that idea. Also, finding out which of the units could appear as Barbarians was nearly impossible. Without access to the source code, there is no way I could have found this information. Trying to put in nuclear weapons was another learning experience. Nuclear weapons (the Infernal Device in the *MoM* scenario) are defined as any unit with an attack strength greater than 98. That was fine, but then I found out that to build nuclear weapons you not only need the technology that allows them, but it's also imperative that the wonder in the Manhattan Project slot must have been built.



of their arrival, then bribe them. I couldn't make them so they couldn't be bribed, unfortunately. Normally, they would have a life span of only so many turns; they can't capture a city, because they're air units, and air units can't occupy a city. So they would normally die in a few turns, as the story has them die of human diseases, but if you're able to capture one, then you can get it into your city and keep it alive. Then you've got this huge unit that goes around and conquers the world, but that takes a lot of the fun out of playing, so I don't recommend it.

Some of the quests were in multiple parts. For instance, the underwater world of Atlantis doesn't appear—it doesn't exist on the map—until somebody defeats Captain Nemo's Nautilus. That creates a specific Iceberg. When the Iceberg is conquered, Atlantis appears. Then, you could go out to Atlantis and discover a technology from the ancient Atlanteans that allows you to move your research forward and get a special advance and a unit. The Great Ape is a similar setup. If you're able to find King Solomon's Mines, then you also get a reference to Skull Island. If you can then get past the reefs, you'll find the home of the Great Ape. Then, if you're able to capture him, he breaks loose and wreaks havoc on New York—you know the story. So you get advantages and sometimes you get penalties for doing some of these things, but it follows a story line, and it makes it interesting.

Eventually there's a new victory condition. Instead of taking a rocket ship to Alpha Centauri, you're able to take a spaceship to the Moon and win that way. Of course, that's only possible if you accomplish a sufficient number of the quests, to get the advances that you need to move along. I was satisfied that I managed to get 8 or 9 different stories—some of my favorite classic science fiction stories—into the game.

Ken Burd

The second add-on pack MicroProse released for *Civilization II* included eleven scenarios, and Mick only designed five of them. Of the rest, Ken Burd—a designer and programmer at MicroProse Texas (formerly SimTex)—designed two. His are based on their hit games *Master of Orion II* and *Master of Magic*. Ken was kind enough to take time out of his busy schedule to write down a few of the pitfalls and problems he ran into during the process.

Master of Magic

Master of Magic (MoM) has a lot in common with *Civilization II*, so when I went into it, I felt that creating a MoM scenario would be a relatively easy task. The strategic situation was the same; it seemed that substituting wonders for spells and having the

are other quests that you can try to complete. They're *all* important to the game, because they're attached to these advances that you need to progress. If you let the other tribes do the quests, you can fall behind, and they begin to research new units and improvements and things that aren't available to you. The theme of this scenario was for civilizations to undertake the general spirit of the time—to explore the world and discover things, apply the newly discovered inventions of science, and become the great world powers that they actually turned out to be.

The trick of using units that look like terrain worked so well in Midgard that I decided to use it in this scenario. What eventually happened, though, was that the testers had so much difficulty locating the quests—because of their invisibility—that I eventually put special icons in all the areas to direct players to them. So, for example, if you're looking for the lost Egyptian city, you'll see several ancient looking cities in Africa, and you have to investigate all of them. Two of them are just red herrings, and one of them is the correct city. You have to try to figure out just by trial and error which is the true city. I did that with King Solomon's Mines, too. There are two or three possible locations for the mines; it varies, because I also put in the same trick of having different starting positions that I used in Samurai. I created, I think, six different starting positions for this scenario. The game randomly picks one, so you're never absolutely sure where all these quests are. In one, you might have to go to Iceland to journey to the center of the earth, and in another one, you might end up having to go into the middle of Siberia before you find the entrance to the center of the earth. In the alternates, I just moved the locations around. I mixed them up, so in another I might have two instead of three ancient cities. I moved the quest locations around so that it wouldn't be completely predictable.

In order to progress, you need to get certain advances, and I tied them in as rewards for accomplishing the quests. For example, if you wanted to get to a certain unit, you might need to do two quests to get the two prerequisite advances to be able to research the next step. So there was a real incentive to go out and explore and have that discoverer's attitude.

I also put in a sort of super-criminal Fu Manchu-type tribe—that was a popular stereotype back then. They were the evil people trying to conquer the world; they're like the Martians. In Edgar Rice Burroughs stories, there was always this evil group of people out to conquer the world. The Secret Evil Society has its own, special set of units. They're really not a player tribe; they're just something that you have to interact with. It's another challenge, and part of the atmosphere.

The Martian invasion was based, of course, on the *War of the Worlds*. At a random turn, the Martians land and invade. They're really powerful units, but actually there's a way to use them. Players have learned to build up this huge treasury in anticipation



It allows you to research in a direction you couldn't normally and get a new unit that is sort of the dominant unit in the game—like the armor unit of the original game. It's a big, powerful ground unit called an Ice Drake.

As the scenario sort of follows the Nordic myths—the Scandinavian myths—I have a final battle that occurs when somebody discovers a certain advance. That triggers Ragnarok, which is the final battle between the elemental giants of the world and the gods. In this, the gods are doomed to lose, and when Ragnarok occurs, all of the forces you're currently able to build disappear, and you're reduced to building one or two units only. The Stygians are destroyed, because they were forces of the underworld god, who is also defeated in the battle. So there's some good and some bad to it, but that's part of the surprise. Also, at the end of the game, there's a new victory condition. If you are able to get to a certain advance, you discover the rainbow bridge that allows you to go to the next plane—Altgard, the one that the gods had inhabited—and you win the game. That's the other way you can win the scenario, besides just conquering the world.

I like putting in surprises that players aren't aware of the first time that they play. That gives them something to discover. In fact, I don't think anybody in QA discovered that conquering a Dragon helped get you to Altgard via the rainbow bridge. In the beginning, I didn't tell the testers anything—to test the surprise factor—but later, of course, I told them about all the features in the game, so that they could test them and make sure they worked.

The World of Jules Verne

The last one was the World of Jules Verne. I hadn't initially intended to do this scenario—it was going to be done by somebody else—but he was too busy and I stepped in. This is the scenario most unlike the original game. In this scenario, I tried to build in a whole series of quests and challenges—outside the normal develop-and-conquer-the-world—that enabled the player to progress and win the game. So, a lot of the advances are keyed specifically to completing these quests. They're sprinkled all over the world, and they're all based on science fiction stories of the nineteenth and early twentieth century. You can discover the lost world, or journey to the center of the earth, or defeat Captain Nemo and the Nautilus.

All these stories that Jules Verne and H.G. Wells and the others like them—H. Rider Haggard comes to mind—told are in the game, plus some real historical adventures that were very famous at the time. Richard Burton's discovery of the source of the Nile and his disguising himself to get into the holy city of Mecca are included. Those

Another feature that I introduced in Atlantis was quests. Special units—monsters—would randomly appear during the game, and if you could go to them and bring enough forces to defeat the monsters, you got a reward—a new advance—and you got a pile of gold to add to your treasury. There were five or six of them in the Atlantis scenario.

The Mythic History of Midgard

Midgard was going to be my big game. I initially intended this to be the scenario that brought together all the new events macro improvements that we made, so I tried a whole lot of things in this one. I had five or six tricks that I tried, things that I thought would enhance the game. I even tried to create underwater worlds and underground tunnel labyrinths.

I guarded the elven capital city with a ring of invisible tree guards. So, if you're one of the other tribes and you're trying to invade the island and get to the capital city, you just accidentally run into these tree guards. Your point units probably get demolished, and you have to decide, "Now what do I do? How do I get past them?" You have to figure out a route and try to build that route into the capital city—to get through these guards, these tree people.

Each of the tribes has special abilities that distinguish them from the other tribes. So, for example, the Eagle people tend to have creatures that can move quickly, and they have more flying creatures. The Elvish are more comfortable in forests, using nature. The tribes also all have slightly different advances under their control at the beginning, so that none of them follows exactly the same research route. We put in a lot of new artwork for this one.

Then there were the portals from the underworld. Those were places where events would randomly generate creatures that would just spread out around the world and attack—skeleton armies and so forth. For the Merfolk tribe, even though I couldn't actually make them build cities in the sea, I tried to disguise some terrain to look like shallow seas that they could use to build their empire. Although they start off weaker, they have more isolation—they have more chance to develop without contact and wars with the other tribes. They turned out to be more of a peaceful tribe.

There are also quests. I expanded on the quests in Midgard, and there are several that occur. You can go after monsters, and if you are able to capture an enemy capital city, it gives you a reward. One of the keys in this scenario is to defeat a Dragon. If you're able to defeat a Dragon, that gives you a special advance that's unavailable elsewhere.



The Atlanteans have one city that's started, but in terms of size, all the civilizations are pretty much starting from scratch except the Europeans. These I meant to represent the precursors of the Celts in northern Europe. They start with a larger population base—more settlers—but they're more primitive; they don't have as many advances as the other tribes. The Atlanteans have certain advances that only they possess, but which they can give away and trade and so forth. The other tribes can't research these. They're all based on the Atlanteans' worship of Neptune, so they're all sea-oriented advances, and they allow special sea-type units. These advances permit communication with sea mammals like porpoises and so forth. The Atlanteans are also slightly more advanced than the other tribes—the Greeks and the Egyptians and the Sumerians.

Then there's the seventh tribe—a special tribe—way up in the corner. This is based on another mythical kingdom—the island of Mu—and the residents, who were called the Lemurians. In the nineteenth century, the existence of this ancient civilization was a very popular idea, because it provided an explanation people could swallow as to how cultures outside of the Middle East and Europe had been able to gain knowledge formerly believed to be the sole province of “civilized” nations. So, for example, that explained the pyramids in Mesoamerica. The hypothesis was that the Lemurians had all of the technologies originally, and then knowledge spread out worldwide from there. That, they believed, was why Mesoamericans could build pyramids and the Egyptians could build pyramids. That explained away a lot of those kind of discrepancies that they couldn't figure out or accept. I made the Lemurians very similar to the Atlanteans, except that they worship the sun. Their disadvantage is they're way off in a corner, and it takes them a while to get going. In this scenario, whatever tribe you're ruling, you need to have contact with both the Atlanteans and the Lemurians. The combination of their advances takes you in a new direction so you can build new units and weapons that were never really developed historically—special tanks and lens-type weapons, and it gets fun.

This was the first scenario in which I tried to build in a really big surprise—an unexpected element. The islands just are destroyed by a volcano. The problem is that, because it's random, it could possibly happen on the first turn. That's when the real shock occurs, because every so often it'll happen very, very quickly. It's actually worse if it happens after you've built up a civilization. It's not perfect game design, because you don't want to punish people for good play—it's kind of counter-productive. In this case, though, it was either that or the islands don't get destroyed—or they get destroyed at a certain time, and that's less interesting—too predictable. Once a player knows that the possibility's there, then they can try to adjust by trying to get onto the mainland.

U.S. \$19.95
Canada \$26.99

Electronic Entertainment

INCLUDES
3 NEW
SCENARIOS
ON CD-ROM

DESIGN AND BUILD YOUR OWN SCENARIOS

Learn from the best! MicroProse insider John Possidente has pried the design secrets out of the leading MicroProse game designers who have contributed *Civilization™II* scenarios. In this unique guidebook, he discusses different sets of scenario-building tools and opens the curtain on the magic and illusion practiced in MicroProse's *Civilization™II* scenarios, including map-making, unit design, game balancing, multiplayer and more. As the acclaimed *Civilization™II* game designer Mick Uhl puts it in his Foreword: "You could do worse than to listen to his advice."

- ▲ Easy step-by-step instructions
- ▲ Single and multiplayer scenario design
- ▲ Favorite secrets and tricks of leading *Civilization™II* scenario designers
- ▲ Clear explanations of the complexities of event and unit construction
- ▲ Creative ways to employ triggers and actions
- ▲ Telling details on units, advances, attributes, City Improvements, Wonders of the World, and Cosmic Principles

MICROPROSE



www.gwpress.com



ISBN 1-56893-904-3



5 1995

